

Ein Ansatz zur zweckmäßigen Aufgabenkonstruktion
für betriebliche Mensch-Computer-Systeme

Dissertation
vorgelegt dem
Fachbereich Rechts- und Wirtschaftswissenschaften der
Johannes Gutenberg Universität Mainz zur Erlangung
des Grades eines Doktors der Staatswissenschaften
(Dr. rer. pol.)

von

Tamnam Adi

1979

Erstberichterstatter: Prof. Dr. H. Kargl

Zweitberichterstatter: Prof. Dr. H. Wedekind

Tag der mündlichen Prüfung: 7. Februar 1979



Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41552622_0002

Ein Ansatz zur zweckmäßigen Aufgabenkonstruktion
für betriebliche Mensch-Computer-Systeme

Dissertation
vorgelegt dem
Fachbereich Rechts- und Wirtschaftswissenschaften der
Johannes Gutenberg Universität Mainz zur Erlangung
des Grades eines Doktors der Staatswissenschaften
(Dr. rer. pol.)

von

Tammam Adi



Erstberichterstatter: Prof. Dr. H. Kargl

Zweitberichterstatter: Prof. Dr. H. Wedekind

Tag der mündlichen Prüfung: 7. Februar 1979

Inhaltsverzeichnis

1. Problemstellung, Zielsetzung und Vorgehensweise der Arbeit.....	1
2. Grundvereinbarungen.....	6
3. Elementare Input-Output-Umwandlungen und Elementarvorschriften..	11
3.1. Eine primitive Input-Output-Umwandlung und eine entsprechende Vorschrift.....	11
3.2. Die Elementarvorschriften.....	12
4. Regeln zur Aufgabenkonstruktion.....	14
4.1. Funktionen als Vorschriften für Datengruppen.....	14
4.2. Grundregeln zur Konstruktion von Vorschriften aus Elementarfunktionen.....	16
4.3. Analyse und Veranschaulichung der Grundregeln.....	18
4.4. Grundgebilde und abgeleitete Gebilde zur hierarchischen Aufgliederung von Aufgaben in Vorschriften.....	25
4.5. Weitere abgeleitete Gebilde.....	29
4.5.1. Verallgemeinerungen für Grundgebilde.....	29
4.5.2. Gebilde mit Ereignissen.....	36
4.6. Routinen.....	41
4.7. Besondere Vorschriften zum Aufbau von Datengruppen.....	43
5. Vergleichende Wertung der ermittelten Aufgabenkonstruktions- gebilde.....	45
6. Die Umwandlung von Konstruktionsgebilden in Befolungsgebilde...	55
7. Praxisnahe Beispiele.....	66
7.1. Die Aufgabenkonstruktion.....	66
7.2. Befolungsstrukturen.....	78
8. Die Erkenntnisse der Arbeit.....	88
Verzeichnis der verwendeten Abkürzungen.....	93
Literaturverzeichnis.....	94

1. Problemstellung, Zielsetzung und Vorgehensweise der Arbeit

Ein betriebliches Mensch-Computer-System sei zunächst als eine Gruppe von Menschen und Computern definiert, die für die Erreichung bestimmter Ziele eines bzw. mehrerer Betriebe arbeiten. Die Erstellung von Aufgaben für ein solches System ist Gegenstand der betriebswirtschaftlichen Organisationslehre. Dort wird eine Aufgabe im Grunde als ein gesetztes Soll betrachtet¹⁾, das durch Menschen und/oder Sachmittel zu erfüllen ist.²⁾ Gehen wir vorläufig von diesem Aufgabenbegriff aus. In der Praxis der Erstellung von Aufgaben für betriebliche Mensch-Computer-Systeme werden in der Regel die fünf folgenden Aufgabenversionen in der angegebenen Reihenfolge erstellt (in Klammern werden bekannte Bezeichnungen angegeben):³⁾

-
- 1) Vgl. Kosiol, E.: Aufgabenanalyse und Aufgabensynthese, in E.Grochla(Hrsg.): Elemente der organisatorischen Gestaltung, Hamburg 1978, S.68
 - 2) Vgl. Witte, E.: Ablauforganisation, in : E.Grochla(Hrsg.): Handwörterbuch der Organisation, Stuttgart 1969, S.20
 - 3) Vgl. Teichroew, D.: Improvements in the System Life Cycle, in: Freeman,P. und Wasserman, A.I.(Hrsg.): Software Design Techniques, Long Beach, California 1976;
Vgl. dazu auch Boehm, B.W.: Software Engineering, in : IEEE Transactions on Computers, Vol. C-25, No.12, Dezember 1976, S.1226 ff.

1. Eine Version aus der Sicht des Betriebes, für dessen Zwecke das geplante System eingesetzt werden sollte (Benutzervorgaben).
2. Eine Version im Hinblick auf das geplante System selbst (Benutzeraufgaben).
3. Eine dem Computereinsatz besonders zugewandte Version (Programmvorgaben).
4. Eine computer-nahe Version (eine Kombination aus Programmen und Aufgaben für Menschen).
5. Eine äußerst computer-nahe Version (eine Kombination aus Maschinenprogrammen und Aufgaben für Menschen).¹⁾

Jede Version wird auf den vorgelagerten Versionen aufgebaut; die Erfüllung der letzten Version durch Menschen und/oder Computer wird solange "ausprobiert", bis "es klappt", wobei alle Versionen verbessert werden.²⁾ Der Arbeitsaufwand für diese Prozedur (sie sei im folgenden Testen genannt) ist in der Regel beinahe genauso groß wie der Arbeitsaufwand für die Er-

-
- 1) Vgl. zu dieser Art der Versionenauffassung auch Hamilton, M. und Zeldin, S.: The Manager as an Abstract Systems Engineer, Technical Report No.5, Higher Order Software, Inc., Cambridge, MA, Juni 1977, S.3 f.
 - 2) Vgl. Boehm, B.W.: Software Engineering, a.a.O. S. 1233 ff. und S. 1226 ff.. Vgl. auch ebenda.

stellung der fünf Aufgabenversionen.¹⁾

Aus der Sicht der betriebswirtschaftlichen Organisationstheorie ist diese Art der Aufgabenerstellung unzweckmäßig; das "fundamentale" Zweckmäßigkeitsprinzip der Aufgabenkonstruktion besagt nämlich, daß die Erfüllung der Aufgaben zu jedem Zeitpunkt der Konstruktion gewährleistet sein sollte.²⁾ In diesem Sinne betrachten wir die obige phasenartige "5-Versionen-dann-Testen"-Vorgehensweise zur Aufgabenkonstruktion als unzweckmäßig. Die konventionelle Betrachtung der Computertechnologie, nach der Betriebe als Technologieprodukt-Anwender und nicht als Technologie-Anwender gelten, ist unseres Erachtens für diese zwischenhändler-mäßige Phasenbetrachtung verantwortlich. Die Verwirrung in der Problematik der Computeranwendung ist auch in diesem Zusammenhang zu sehen.³⁾

Unter diesen Umständen ist es angebracht, mit dem obigen Verständnis der Konstruktionszweckmäßigkeit eine abstrakte grundlagenmäßige Analyse der Aufgabenbeschaffenheit in einem betrieblichen Mensch-Computer-System durchzuführen.

- 1) Vgl. Boehm., B.W.: Software Engineering, a.a.O., S.1233
Vgl. Hamilton, M. und Zeldin, S.: Higher Order Software- A Methodology for Defining Software, in: IEEE Transactions on Software Engineering, Vol. SE-2, No. 1, März 1976, S.9
- 2) Vgl. Kosiol, E.: Aufbauorganisation, in: E.Grochla(Hrsg.): Handwörterbuch der Organisation, Stuttgart 1969, S.178 f.
- 3) Vgl. Hamilton, M.u.a.: The Manager..., a.a.O., S.5 f. sowie Teichroew, D.: Improvements..., a.a.O., S. 37, insbesondere folgende Aussage: "There are many differing opinions on...what are symptoms rather than problems..."

Aufgrund der obigen Überlegungen formulieren wir die folgende Arbeitsaufgabe :

Finde einen Aufgabenbegriff für betriebliche Mensch-Computer-Systeme und eine entsprechende Aufgabenkonstruktionsweise, die zweckmäßig ist.

Wir werden wie folgt vorgehen :

Zunächst werden wir Grundvereinbarungen treffen, die den gesuchten Aufgabenbegriff umschreiben und die zu entwickelnde Aufgabenkonstruktionsweise grundlegend definieren. (Kapitel 2)

Anschließend werden Elementarteilchen für Aufgaben der definierten Art ermittelt. (Kapitel 3)

In Kapitel 4 wird der Zusammenhang solcher Elementarteilchen untersucht, um Regeln für die Konstruktion von Aufgaben aus ihren Elementarteilchen zu ermitteln. Dabei werden anschauliche Konstruktionsgebilde entwickelt.

Aufgrund der Erkenntnisse dieses Kapitels werden wir dann zu einigen bekannten Gebilden zur Aufgabenkonstruktion Stellung nehmen. (Kapitel 5)

Anschließend wird eine Vorgehensweise zur Umwandlung von Aufgabenstrukturen in Aufgabenerfüllungsstrukturen (sogenannte Befolungsstrukturen) eingeführt. (Kapitel 6)

In Kapitel 7 werden praxisnahe Beispiele für die Aufgabenkonstruktion und die Umwandlung von Aufgabenstrukturen in Befolungsstrukturen dargestellt.

Schließlich werden die Erkenntnisse der Arbeit systematisch bewertet. (Kapitel 8)

2. Grundvereinbarungen

Ein Objekt ist etwas Wahrnehmbares (z.B. ein Mensch, ein Buch, eine Handlung oder ein Gedanke). Der Mensch unterscheidet sich von den übrigen Objekten dadurch, daß er einem beliebigen Objekt ein aus beliebigen Objekten bestehendes Symbol so zuordnen kann, daß eine gedankliche Verbindung zwischen Objekt und Symbol hergestellt wird, ohne daß diese Zuordnung bzw. Verbindung von Vorschriften geleitet wird. Diese Fähigkeit nennen wir Erstbenennungsvermögen. Da Vorschriften selbst aus Symbolen bestehen, ist es Menschen vorbehalten, Vorschriften zu kreieren.

Im folgenden gehen wir von der nun anzugebenden präzisen Definition des Begriffes 'Vorschrift' aus: Eine Vorschrift ist eine Symbolkombination, die mit einer Umwandlung eines Objektes (Input genannt) in ein Objekt (Output genannt) (kurz: mit einer I-O-Umwandlung) von einem Menschen gedanklich und/oder durch ein Sachmittel so verbunden werden kann, daß diese Umwandlung durch Menschen und/oder Sachmittel ohne weiteres vollzogen werden kann. Diesen Vollzug nennen wir Vorschriftenbefolgung. Ist die genannte Verbindung für diesen Vollzug nicht ausreichend, so sprechen wir nicht von einer Vorschrift, sondern von einer Aufgabe. (Siehe die folgenden Abbildungen)

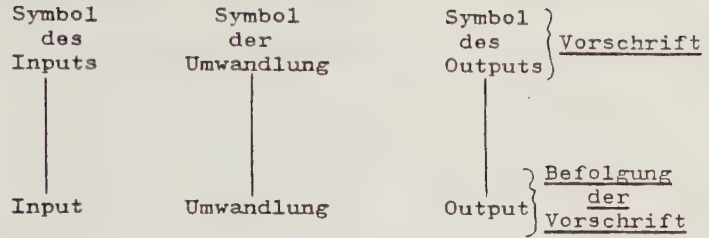


Abb.1. Veranschaulichung einer Vorschrift

Die veranschaulichte Vorschrift ist insofern einfach, als die Symbolkombination eine einfache Symbolfolge und die Verbindungen zu den Objekten (Input, Umwandlung und Output) getrennt und einfach sind. Die folgende Abbildung veranschaulicht eine Aufgabe; die eben genannten Verbindungen sind hier für eine "Befolgung" nicht ausreichend (gestrichelt!); die Objekte werden gesucht.¹⁾

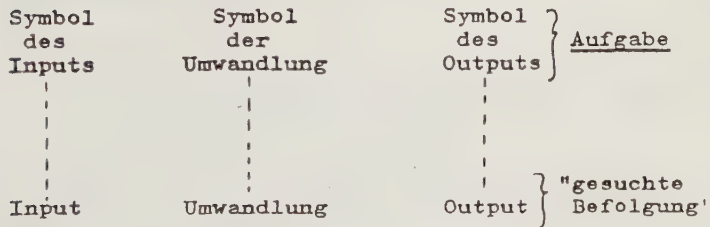


Abb.2. Veranschaulichung einer Aufgabe

1) Vgl. den Aufgabenbegriff in:
 Handlexikon Organisation, herausgegeben von
 der Akademie für Organisation in Gießen,
 Frankfurt am Main 1976;
 Witte, E.: Ablauforganisation, a.a.O., S.21 ff.
 Dort wird eine Aufgabe als eine Art I-O-Um-
 wandlungssoll definiert.

Symbole und Symbolkombinationen, die selbst Inputs bzw. Outputs von Vorschriftenbefolgungen (d.h. von I-O-Umwandlungen) sind, sollen im folgenden Daten heißen. Symbolisieren Daten Objekte und/oder Objektkombinationen von Betrieben, so heißen sie betriebliche Daten.

Ein Sachmittel zur Umwandlung von Daten(Input) in Daten(Output) heißt Datenmaschine. Beispiele für Datenmaschinen sind Zentraleinheiten von Computern, Locher, Drucker, Magnetbandeinheiten, Datenübertragungseinheiten etc.

Wir präzisieren nun den in der Arbeitseinleitung gegebenen Begriff des betrieblichen Mensch-Computer-Systems :

Ein betriebliches Mensch-Computer-System ist eine Gruppe von

- a) Menschen, die Aufgaben kreieren, Aufgaben in Vorschriften umwandeln und Vorschriften befolgen und
- b) Datenmaschinen, die Vorschriften befolgen, wobei als Inputs und Outputs nur betriebliche Daten(von einem bzw.mehreren Betrieben) in Frage kommen.¹⁾

1) Die Kreation von Vorschriften kann als Kreation von Aufgaben und ihre Umwandlung in Vorschriften aufgefaßt werden.

Im folgenden verstehen wir unter Inputs und Outputs ausschließlich betriebliche Daten. Wir gehen davon aus, daß man für jede Vorschrift eine Datenmaschine konstruieren kann, die sie durch Abtasten und darauf Reagieren befolgen kann.¹⁾

Wir gehen ferner davon aus, daß man in einem Mensch-Computer-System grundsätzlich anstrebt, daß alle Vorschriften durch Datenmaschinen (und nicht durch Menschen) befolgt werden.²⁾ Dem Begriff der Vorschrift³⁾ und diesen beiden Vereinbarungen entnehmen wir die folgende Feststellung:

Eine Vorschrift gilt nur dann als solche, wenn sie zweckmäßig aufgebaut⁴⁾ ist und somit ohne Testen⁵⁾ durch eine Datenmaschine befolgt werden kann.⁶⁾

-
- 1) Vgl. dazu Brauer, W. und Indermark, K.: Algorithmen, rekursive Funktionen und formale Sprachen, Mannheim 1968, S. 7
 - 2) Wir gehen davon aus, daß reine Vorschriftenbefolgung sowohl unwirtschaftlich als auch unmenschlich ist (wenn sie von Menschen vollzogen wird).
 - 3) Vgl. S. 6
 - 4) Vgl. S. 3
 - 5) Vgl. S. 2
 - 6) Die Datenmaschine muß nicht tatsächlich existieren, sondern als mit Sicherheit konstruierbar gelten (etwa aufgrund eines mathematischen Beweises).

Die Menschen in einem betrieblichen Mensch-Computer-System tun also in der Regel nichts anderes als Aufgaben und/oder Vorschriften für Datenmaschinen zu kreieren. Dabei könnten sie im Sinne von übergeordneten Aufgaben handeln, d.h. diese Aufgaben in Vorschriften umzuwandeln versuchen.

Die Kreation von Aufgaben ist in der Regel eine einfache Handlung; sie könnte nur darin bestehen, ein- bzw. mehrfach drei Symbole für Input, Umwandlung und Output anzugeben, ohne ihre Verbindungen zu den jeweiligen Objekten zu definieren.¹⁾

Deshalb betrachten wir im folgenden die Umwandlung von Aufgaben in Vorschriften (d.h. die Vervollständigung von Aufgaben zu Vorschriften) als die eigentliche Aufgabenkonstruktionsarbeit.

1) Es reicht unter Umständen aus, ein Output-symbol anzugeben. Vgl. dazu Witte, E.: Ablauforganisation, a.a.O., S.21

3. Elementare Input-Output-Umwandlungen und Elementarvorschriften

Aufgaben und Vorschriften sind im Grunde nichts anderes als Symbole für I-O-Umwandlungen. Deshalb wollen wir zunächst die Beschaffenheit von I-O-Umwandlungen untersuchen.

Manche Naturwissenschaftler suchen nach den Elementarteilchen der Objekte, um ihre Zusammensetzung (Struktur) kennenzulernen und aufgrund dessen neue Objekte konstruieren zu können. In diesem Kapitel wollen wir analog Elementarteilchen für die Objektgruppe 'I-O-Umwandlungen' (elementare I-O-Umwandlungen genannt) bestimmen und aufgrund dessen Elementarvorschriften definieren, deren Strukturierung in Vorschriften später untersucht wird, um eine Grundlage für die zweckmäßige Aufgabenkonstruktion zu schaffen.

3.1. Eine primitive Input-Output-Umwandlung und eine entsprechende Vorschrift

Die Zahl 5 sei ein betriebliches Datum, dessen Umwandlung in die Zahl 6 erwünscht wird. Es gibt mit Sicherheit eine Datenmaschine (z.B. Tischrechenmaschine), die diese Umwandlung vollziehen kann, indem sie die Vorschrift

$$5 \quad +1 \quad 6$$

befolgt.

5 und 6 sind Symbole für einzelne Symbole (Daten, die nicht auf diesem Papier sind).

Vorschriften, deren Befolgung jeweils die Umwandlung eines Symbols (Inputdatum) in ein Symbol (Outputdatum) bedeutet, symbolisieren wir jeweils durch

$$x \rightarrow y$$

und nennen sie primitive Vorschriften (für primitive I-O-Umwandlungen).

3.2. Elementarvorschriften

Turingmaschinen sind Idealisierungen für Datenmaschinen.¹⁾ Turing-berechenbare Algorithmen, normale Markov-Algorithmen und rekursive Funktionen sind Idealisierungen für Vorschriften.²⁾ Für jeden dieser drei Begriffe gilt bisher die These, daß jede Vorschrift ihm unterliegt, und man hat bewiesen, daß die drei Begriffe äquivalent sind³⁾.

1) Vgl Brauer, W.u.a.: Algorithmen..., a.a.O., S.7

2) Vgl. ebenda, S.4 ff.

3) Vgl. ebenda, S.4 ff. und S.94

Aufgrund dieser Überlegungen haben wir aus der Literatur zu diesen drei Vorschriftsbegriffen die folgende Feststellung ableiten können:¹⁾

Alle Vorschriften sind Kombinationen aus den drei folgenden primitiven Vorschriften, die wir somit Elementarvorschriften nennen:

- 1) Konstantenerzeugung: Ein Input wird in ein festgelegtes Symbol (Konstante, K) umgewandelt:

$$x \rightarrow K$$

(z.B. $x \rightarrow 0$ oder $x \rightarrow 1$)

- 2) Duplizierung: Ein Input wird in einen mit ihm identischen Output umgewandelt:

$$x \rightarrow \underline{x}$$

Die Unterstreichung dient der Unterscheidung zwischen Input und Output.

- 3) Nachfolgerangabe: Sind zwei Symbole (als ein Symbol betrachtet) als Input vorgegeben, so wird das rechts liegende Symbol (der Nachfolger) als Output angegeben:

$$x_1 x_2 \rightarrow x_2$$

1) Vgl. Brauer, W. u.a.: Algorithmen..., a.a.O., S.69 ff.
Vgl. Heidler, K.; Hermes, H. und Mahn, F.-K.: Rekursive Funktionen, Zürich 1977, S.94

4. Regeln zur Aufgabenkonstruktion

4.1. Funktionen als Elementarvorschriften bzw. Vorschriften für Datengruppen

Eine (mathematische) Funktion ist eine 'Vorschrift' die jedem Datum aus einer ersten Gruppe (Inputgruppe) ein Datum aus einer zweiten Gruppe (Outputgruppe) zuordnet.¹⁾ Dabei kann diese 'Vorschrift' sowohl eine Aufgabe als auch eine (in unserem Sinne) echte Vorschrift sein; im zweiten Falle spricht man von einer rekursiven Funktion; nur in diesem Falle ist sie für uns von Nutzen; im folgenden verstehen wir unter Funktion ausschließlich eine rekursive Funktion und fassen sie als eine Vorschrift für Datengruppen auf.²⁾

Eine Datengruppe symbolisieren wir durch Indizierung eines Buchstabens (z.B. x_1 , x_2 , wobei x bzw. x_k ein beliebiges Datum und X die gesamte Datengruppe symbolisieren; X_1 , $X_2 \dots$ etc. symbolisieren Teilgruppen aus X ; L ist eine leere Teilgruppe bzw. Datengruppe). Besteht jedes Datum aus mehreren Symbolen, so benutzen wir mehrere durch Kommata getrennte Buchstaben (z.B. $(x, w); (x_k, w_k); (X, W); (X_1, W_1), (X_2, W_2)$).

1) Vgl. Meschkowski, H.: Mathematisches Begriffswörterbuch, Mannheim 1976, S. 13. Dort spricht man von Mengen anstatt Gruppen; wir bevorzugen jedoch den anschaulichen Begriff der Gruppe.

2) Vgl. Brauer, W. u. a.: Algorithmen..., a. a. O., S. 69 ff.

Wir schreiben

$$A = B \text{ (bzw. } A \neq B \text{)}$$

und meinen damit, daß A zur Zeit nichts anderes (bzw. etwas anderes) ist als B, wobei A und B beliebige Symbole, Daten bzw. Datengruppen sein können. Aufgaben für Datengruppen symbolisieren wir durch

$$A_i(x) = y, \quad A_i(x_k) = y_k \quad \text{bzw.}$$

$$A_i(X) = Y$$

und meinen damit, daß die Aufgabe mit der Nummer i jedem x aus X (Inputgruppe) ein y aus Y (Outputgruppe) zuordnen sollte. Handelt es sich dabei um Vorschriften für Datengruppen (Funktionen), so schreiben wir V_i anstatt A_i . Die folgenden Funktionen entsprechen den Elementarvorschriften (auf Datengruppen wiederholt angewandt) und heißen deshalb Elementarfunktionen:

$$V_1(x) = K \quad (\text{Konstantenerzeugung})$$

$$V_2(x) = x \quad (\text{Duplizierung})$$

$$V_3(x_k) = x_{k+1} \quad (\text{Nachfolgerangabe})^1)$$

Dabei besteht die Outputgruppe von V_1 aus einer

1) Vgl. Brauer, W.u.a.: Algorithmen..., a.a.O., S.69. Bei der Nachfolgerangabe-Funktion gehen wir davon aus, daß die Daten in der Inputgruppe wie folgt nach Nummern angeordnet sind: $x_1, x_2, \dots, x_k, x_{k+1}, \dots$ etc.

Wir geben x_k als Input an, obwohl eine Datenmaschine sowohl x_k als auch x_{k+1} jeweils als Input "bekommt", wenn sie diese¹⁾ Vorschrift befolgen soll.

Konstanten (bzw. aus Daten, die alle mit dieser Konstanten identisch sind ; jedem Input wird eine von ihnen als Output zugeordnet). Die Outputgruppe von V_2 ist ein Duplikat der Inputgruppe. Die Outputgruppe von V_3 ist auch ein Duplikat der Inputgruppe, hier fehlt jedoch der Input mit der kleinsten Nummer($\underline{x}_1$). Der Nachfolger des Inputs mit der größten Nummer ist nicht vorhanden (wir sagen: aus L).

4.2. Grundregeln zur Konstruktion von Vorschriften aus Elementarfunktionen

In der Praxis werden nur selten Vorschriften für einen Input (und mit einem Output) konstruiert. Daten kommen in Gruppen vor (Dateien) oder können aufgrund der gemeinsamen bzw. ähnlichen I-O-Umwandlungen in Gruppen zusammengefaßt werden. Deshalb behandeln wir im folgenden nur Vorschriften , die aus Elementarfunktionen bestehen. Jede Vorschrift besteht aus Elementarfunktionen, deren Zusammenhang einer bzw. mehrerer der folgenden Regeln unterliegen muß: ¹⁾

A) Kopplungsregel: A_1 entsteht durch Kopplung

1) Vgl. Brauer, W. u.a.: Algorithmen..., a.a.O., S.69 ff. Bei der Erklärung der Regeln benutzen wir Aufgaben(A_i). Diese sind entweder durch Elementarfunktionen oder durch diesen Regeln unterliegende Kombinationen aus ihnen zu ersetzen. Erst danach dürfen wir von Vorschriften sprechen.

von A_2 und A_3 (A_2 wird mit A_3 gekoppelt) wie folgt:

$$A_2(x)=z \quad , \quad A_3(z)=y \quad : \quad A_1(x)=y$$

z heißt Zwischendatum.

- B) Verkettung: A_1 entsteht durch eine Verkettung mit der Startaufgabe A_2 und der Hauptaufgabe A_3 wie folgt:

$$A_2(x_1)=y_1 \quad , \quad A_3(x_k, x_{k-1}, y_{k-1})=y_k \quad : \quad A_1(x)=y$$

- C) Minimalisierung: A_1 entsteht durch Minimalisierung von A_2 wie folgt:

$$A_2(x, y)=z \quad : \quad A_1(x)=y_{\min}$$

wobei A_1 jedem gegebenen Input x denjenigen Output y zuordnet, der (zusammen mit x) als Input von A_2 die kleinste Nummer ($y_{\min}$!) unter denen hat, für die z minimal (etwa $z=0$) wird.

Wir haben die mathematischen Namen der ersten beiden Regeln (Komposition und primitive Rekursion) durch anschauliche Begriffe ersetzt. Im nächsten Abschnitt werden wir diese Regeln analysieren und anschaulich machen, um eine Grundlage für die zweckmäßige Aufgabenkonstruktion zu schaffen.

4.3. Analyse und Veranschaulichung der Grundregeln

Die vorliegende Arbeit ist hierarchisch auf-
gegliedert. Wir könnten eine Baumdarstellung
dafür konstruieren, um den Inhalt unserer Arbeit
anschaulich zu machen :

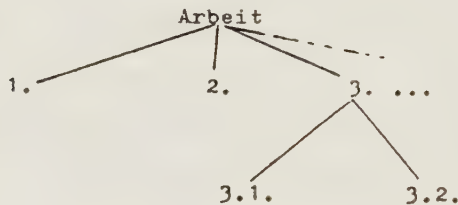


Abb.3. Baumdarstellung für die Arbeit

Eine solche Darstellung ist übersichtlich, trans-
parent, allgemein anerkannt und uralt. Es gibt
"Bäume" für Kirchenhierarchien, die Abstam-
mung von Menschen, den Aufbau von Organisationen
und das menschliche Nervensystem.

Bei der Analyse der Grundregeln von Abschnitt 4.2
ist uns eine hierarchische Aufbauweise aufgefal-
len.¹⁾ Die erklärenden Aufgaben (A_2 und A_3) können
unter Anwendung derselben Regeln mehrfach (hier-
archisch) ersetzt werden :

1) Vgl. Fußnote 1, S.16

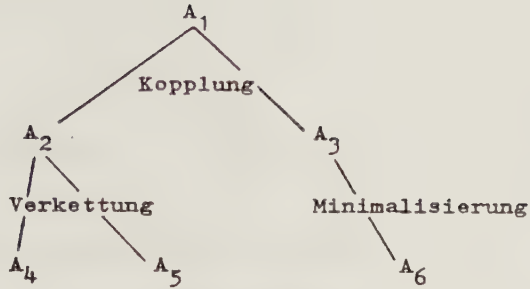


Abb.4. Beispiel zum hierarchischen Aufbau
der Regelanwendung

Die ersten beiden Regeln sind selbst hierarchisch aufgebaut :

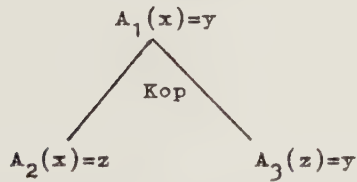


Abb.5. Das Kop-Gebilde: der hierarchische Aufbau
der Kopplung

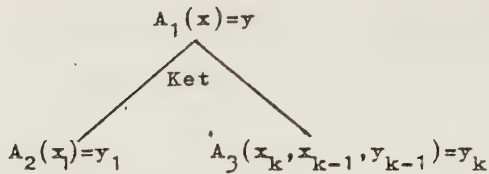


Abb.6. Das Ket-Gebilde: der hierarchische Aufbau der Verkettung

Die entsprechenden Gebilde (siehe Abbildungen) nennen wir Kop- und Ket-Gebilde.

Das Ket-Gebilde repräsentiert für den Fall, daß A_2 und A_3 (Start- und Hauptaufgabe) durch Elementarvorschriften ersetzt werden, eine einfache Kette von elementaren I-O-Umwandlungen: ¹⁾

$$\begin{array}{l}
 x_1 \longrightarrow y_1 \\
 x_2, x_1, y_1 \longrightarrow y_2 \\
 \vdots \\
 \vdots
 \end{array}$$

Die Länge dieser Kette ist für jede Befolgung der Vorschrift, die A_1 entspricht, von der Nummer des jeweiligen Inputs x abhängig.

Bei der Analyse der Minimalisierung konnten wir das folgende Gebilde entwickeln :

-
- 1) Jede I-O-Umwandlung baut auf der "Vorgängerin" auf, indem sie ihren Input und ihren Output als Input benutzt. Bei der Kopplung ist dagegen das Zwischendatum einziger Input der einen (A_3) und einziger Output der anderen (A_2) Aufgabe.

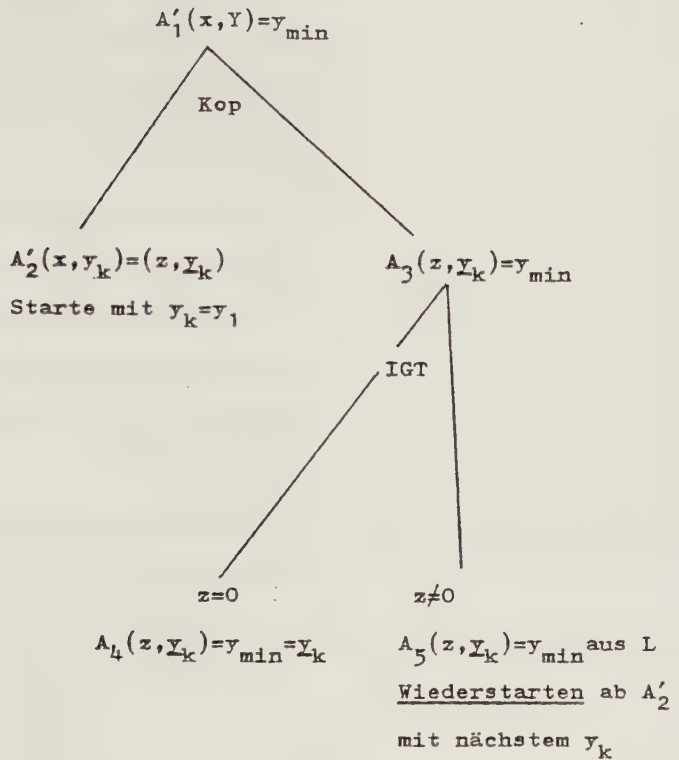


Abb.7. Gebilde der Minimalisierung als Kombi-
nation aus drei Gebilden: einer Kopplung, einer
Inputgruppenteilung(IGT) und einem Wiederstart-
gebilde

Die Minimalisierung umfaßt drei Gebilde: ¹⁾

- 1) Eine Kopplung.
- 2) Eine Inputgruppenteilung(IGT):

Die Inputgruppe von A_3 wird in zwei Teilgruppen:

a) die Paare (z, y_k) mit $z=0$ und

b) die Paare (z, y_k) mit $z \neq 0$

aufgespalten, wobei es keine Überlappung (gemeinsames Paar) geben kann. Jeder Teilgruppe wird eine Aufgabe (A_4 bzw. A_5) zugeordnet.

- 3) Ein Wiederstartgebilde:

Nach jeder Befolgung der 'Aufgabe' A_5 (es handelt sich um eine Konstantenerzeugung, die keine Aufgabe, sondern eine Elementarfunktion darstellt) startet man ab A'_2 mit dem nächsten Input (y_{k+1} z.B.) wieder. A_5 ist einer IGT zugeordnet; A'_2 ist einer in der Hierarchie direkt vorgelagerten Kopplung zugeordnet.

Die Aufgabe A'_1 hat als Input jeweils ein x und die gesamte Gruppe Y . A'_2 entspricht A_2 ²⁾ und einer Duplizierungsfunktion für y_k . Die Vorschrift, die A'_2 ersetzen sollte, wird zunächst mit y_1 , dann $y_2 \dots$ etc. solange wiederholt (befolgt), bis $z=0$ gilt. Dann ordnet A_4 dem jeweiligen x -

1) Wir sprechen von Gebilden als Regeln und umgekehrt.

2) Vgl. die Minimalisierungsregel, S.17

y_k -Paar den Teilinput y_k (unter dem Namen $y_{\min}$) als Output zu.

Dieser rückkopplungsartiger Probiervorgang¹⁾ soll den Namen Testschleife erhalten. Er definiert hier die in der Minimalisierungsregel angegebene Aufgabe $A_1(x)=y_{\min}$ anhand

- a) einer Kopplung zur Weitergabe von z und y_k an einer Aufgabe A_3 , die durch ein
- b) Entscheidungsgebilde (IGT) erklärt wird. Diese Entscheidungsaufgabe bestimmt aufgrund von z , ob A_2' mit einem neuen Input (etwa x, y_{k+1})
- c) wiedergestartet werden sollte (in diesem Falle ist $y_{\min}$ noch unbekannt-- aus L) .²⁾

Wir betrachten die Minimalisierung also als eine besondere Testschleife, die u.a. zwei Grundgebilde definiert (Entscheidungs- und Wiederstartgebilde), die wir als die eigentlichen durch die Minimalisierungsregel gegebenen Grundregeln zur Konstruktion von Vorschriften aus Elementarfunktionen betrachten.

Diese Erkenntnisse haben uns motiviert, die im Gegensatz zur Kopplung nicht einfach anmutende Verkettung zu analysieren. Wir konnten die ers-

-
- 1) Gemeint ist der der Minimalisierung zugrunde liegende Vorgang.
 - 2) Es erfolgt kein Wiederstarten mehr, sobald das y mit der größten Nummer k erreicht wird.

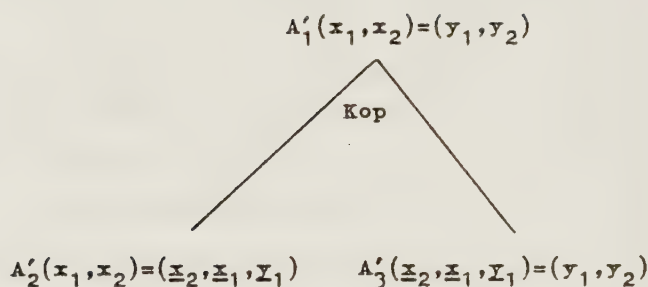


Abb.8. Die ersten zwei Schritte einer Verkettung als Kopplung. In Verallgemeinerung dieses Prinzips gilt die Verkettung als eine Kopplungskette mit variabler Länge.

ten zwei Schritte der Verkettung ¹⁾ mit Hilfe der Duplizierung als Kopplung erklären. Die restlichen Schritte können analog erklärt werden. Somit erscheint die Verkettung als Kopplungskette mit variabler Länge. ²⁾

1) Vgl. S.20

2) Vgl. ebenda

4.4. Grundgebilde und abgeleitete Gebilde zur hierarchischen Aufgliederung von Aufgaben in Vorschriften

Aufgrund der Erkenntnisse von Abschnitt 4.3 machen wir die folgenden Feststellungen:

- 1) Als Grundgebilde zur Konstruktion von Vorschriften aus Elementarfunktionen betrachten wir die Kopplung, die Inputgruppenteilung (Entscheidungsgebilde) und das Wiederstartgebilde. Außerdem führen wir zur Zusammenfassung zweier unabhängiger Aufgaben (bzw. zur Spaltung einer Aufgabe in zwei unabhängige Aufgaben) das folgende Gebilde (einfache Spaltung (ES) genannt) ein:¹⁾

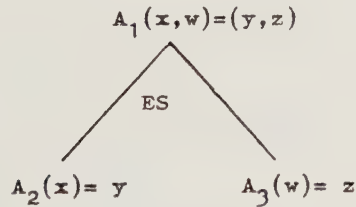


Abb.9. Einfache Spaltung (ES)

- 1) A_2 und A_3 sind unabhängige Aufgaben, da sie keine gemeinsamen Daten haben und deshalb getrennt (als Vorschriften) befolgt werden können.

Wir betrachten die einfache Spaltung auch als Grundgebilde und sprechen im folgenden von den vier Grundgebilden.

- 2) Die Minimalisierung und die Verkettung können aus den Grundgebilden konstruiert werden und heißen deshalb abgeleitete Gebilde; ihnen entsprechen die folgenden Figuren, die sie zwar 'zusammengefaßt' wiedergeben, mit den vollständigen Gebilden ¹⁾ jedoch ohne weiteres verbunden werden können: ²⁾

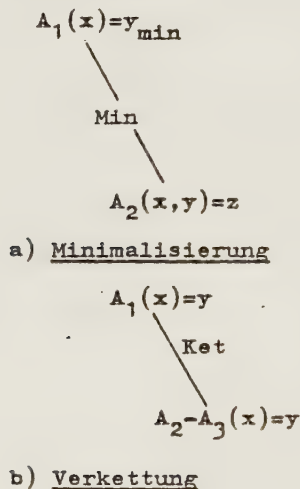


Abb.10. Zwei abgeleitete Gebilde

-
- 1) Vgl. S.20 f. und S.24
- 2) Bei der Minimalisierung könnten z.B. A_1 und A_2 den Aufgaben A'_1 und A'_2 zugeordnet werden.

Wird die Minimalisierung verallgemeinert und als Testschleife aufgefaßt, so entspricht ihr das folgende abgeleitete Gebilde:¹⁾

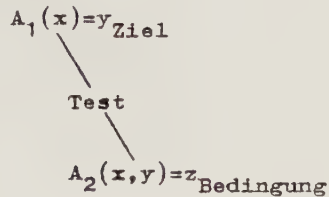


Abb.11. Die Testschleife als abgeleitetes Gebilde

y_{Ziel} wird hier aufgrund der Ergebnisse einer Hilfsaufgabe A_2 in einem Probiervorgang gesucht.

- 3) Unter Anwendung der vier Grundgebilde und (aus ihnen konstruierter) abgeleiteter Gebilde kann man eine Aufgabe hierarchisch in Vor-schriften aufgliedern: Jede Aufgabe wird so lange durch weitere Aufgaben erklärt, bis man Elementarfunktionen erhält (durch V symbolisiert, siehe Abbildung 12).

1) Vgl. S.23

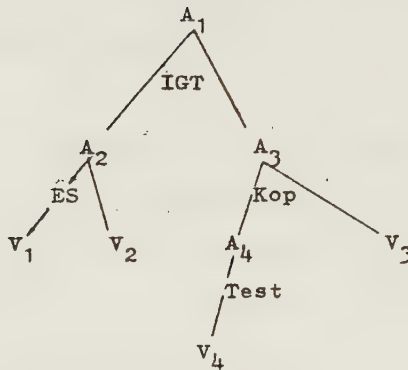


Abb.12. Beispiel zur hierarchischen Aufgliederung von Aufgaben in Vorschriften mit Hilfe von Gebilden

- 4) Man kann weitere abgeleitete Gebilde aus den Grundgebilden konstruieren, die jeweils in ihren zusammengefaßten Formen benutzt werden können. Das werden wir in Abschnitt 4.5 tun.
- 5) Elementarfunktionen können anhand der Gebilde zu häufig verwendbaren Routinen zusammengefaßt werden, die in der hierarchischen Aufgliederung wie Elementarfunktionen eingesetzt werden können. In der Literatur zu den rekursiven Funktionen sind ähnliche Konstruktionen gemacht worden. In Abschnitt 4.6 werden solche Zusammenfassungen (Routinen) behandelt.

4.5. Weitere abgeleitete Gebilde

4.5.1. Verallgemeinerungen für Grundgebilde

A) Allgemeine Kopplung(AKop):

Zwei Aufgaben A_I und A_{II} werden so 'gekoppelt', daß sie jeweils einen Teilinput (x_I bzw. x_{II}) der durch diese 'Kopplung' definierten Aufgabe A_1 ($x=(x_I, x_{II})$) als Input benutzen können: ¹⁾

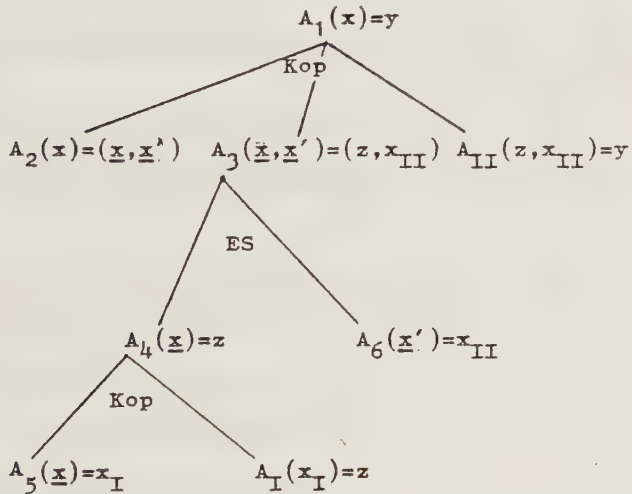


Abb.13. Allgemeine Kopplung, vollständiges Gebilde

1) Vgl. Hamilton, M. und Zeldin, S: The Manager..., a.a.O., S.8 zur Grundidee dieses und der folgenden abgeleiteten Gebilde AES und AIGT.

x_I und x_{II} können sich überlappen. Wir benutzen hier die römische Numerierung (Vgl. S.14) anstatt verschiedener Buchstaben für Teilsymbole.

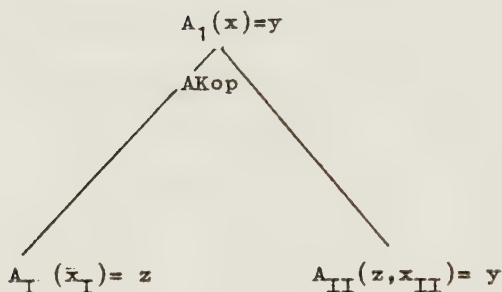


Abb. 14. Allgemeine Kopplung(zusammengefaßt)

In Abbildung 13 kommen Aufgaben vor, die der zweifachen(A_2) bzw. Teilinputduplizierung(A_5, A_6) dienen.

B) Allgemeine Inputgruppenteilung(AIGT):

Zwei Aufgaben A_I und A_{II} werden durch eine Art Inputgruppenteilung zu einer Aufgabe A_1 so zusammengefaßt, daß sie jeweils einen Teil input von A_1 (Symbole wie oben unter B) als Input beanspruchen können:

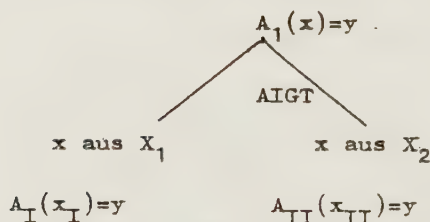


Abb. 15. Allgemeine Inputgruppenteilung(AIGT), (zusammengefaßt).

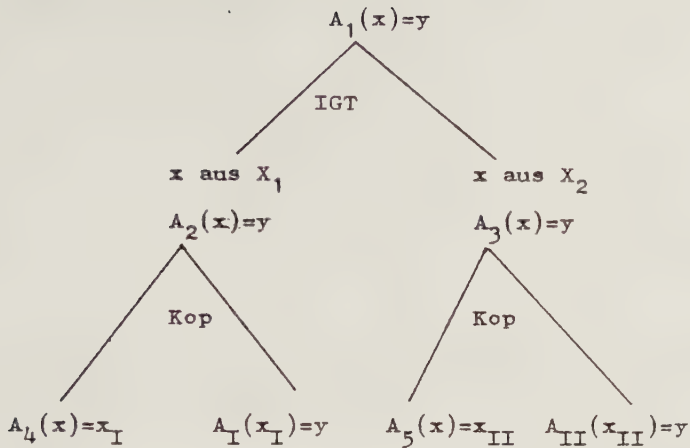


Abb.16. Allgemeine Inputgruppenteilung (vollständig)

Die Inputgruppe von A_1 wird hier in zwei sich nicht überlappende Teilgruppen X_1 und X_2 aufgespaltet. Die allgemeine Inputgruppenteilung kann als allgemeines Entscheidungsgebilde betrachtet werden.¹⁾

1) Vgl. S.22 f.

C) Allgemeine einfache Spaltung(AES):

Eine Aufgabe A_1 wird in zwei unabhängige

Aufgaben A_I und A_{II} so gespaltet, daß letzte-

re Teilinputs von A_1 (Symbole wie oben unter A

als Input haben:

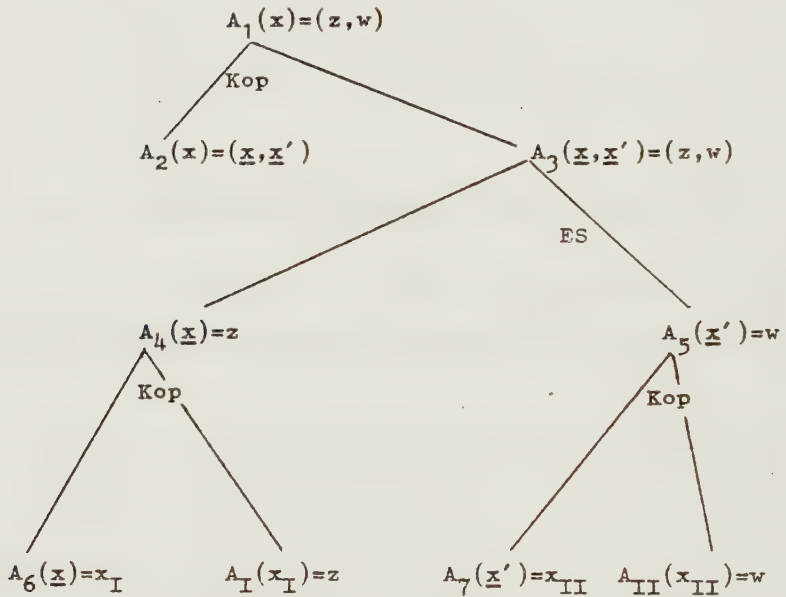


Abb.17. Allgemeine einfache Spaltung(vollständig)

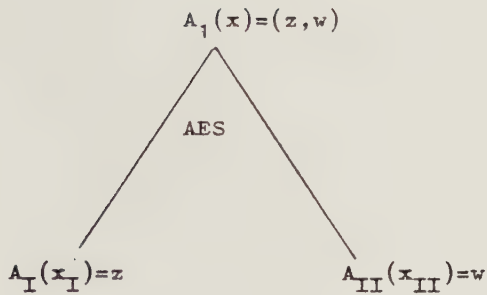


Abb.18. Allgemeine einfache Spaltung
(zusammengefaßt)

D) Allgemeines Wiederstartgebilde:

Beim Wiederstartgebilde erfolgt ein Sprung "nach oben" (in der Hierarchie) von einer Inputgruppenteilung(Entscheidungsgebilde) nach einer Kopplung, die unmittelbar vorge-lagert ist. Dieser Sprung erfolgt hier von einer allgemeinen Inputgruppenteilung(allgemeinem Entscheidungsgebilde) nach einer allgemeinen Kopplung. Dabei kommt es darauf an, daß ein Teilinput (x) der ersten Aufgabe in AKop (A_2) nach der Befolgung der Vorschrift, die einer Aufgabe von AIGT entspricht, neu bestimmt wird (man kann u.U. denselben Teilinput x nochmals einsetzen) und daß das Wie-

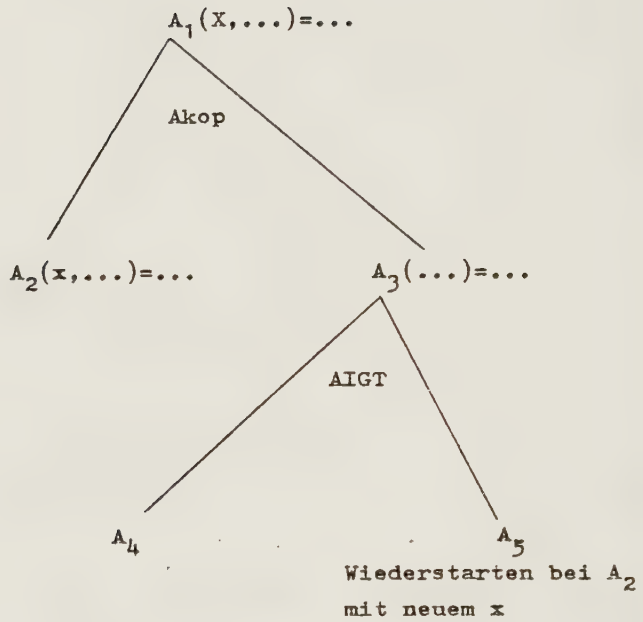


Abb.19. Allgemeines Wiederstartgebilde

derstarten mit Sicherheit irgendwann nicht mehr stattfinden kann. Bei dem (einfachen) Wiederstartgebilde in der Minimalisierung war die Abbruchbedingung das Ausschöpfen von X .

Wir können einen Teilinput (x_I bzw. x_{II}) so auffassen, daß er

- a) nichts umfaßt (aus L) ,
- b) den ganzen Input umfaßt (z.B. $x_I = x$) oder
- c) sich nicht mit dem anderen überlappt.

Aufgrund dieser Überlegungen kann jedes Grundgebilde als Sonderfall des jeweiligen allgemeinen Gebildes aufgefaßt werden, z.B.:

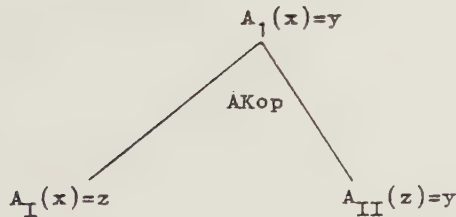


Abb.20. Grundgebilde als Sonderfall des allgemeinen Gebildes , $x_I = x$; x_{II} aus L .

Der in der Abbildung veranschaulichte Sonderfall der allgemeinen Kopplung ist nichts anderes als eine (einfache) Kopplung.

4.5.2. Gebilde mit Ereignissen

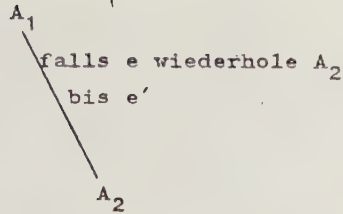
Das Auftreten eines besonderen Inputs bzw. Outputs kann als (von Menschen beobachtbares oder durch Datenmaschinen abtastbares) Ereignis (Symbol : e) aufgefaßt werden. Menschen können aufgrund solcher Ereignisse in die Vorschriftenbefolgung (durch andere Menschen und/oder Datenmaschinen) eingreifen und Änderungen (Umgestaltung des Vorschriftenaufbaus, 'Diktieren' von Inputs und Outputs, Wiederstarten bei bestimmten Vorschriften ... etc.) bewirken.¹⁾

Im folgenden werden einige abgeleitete Gebilde mit Ereignissen dargelegt.²⁾

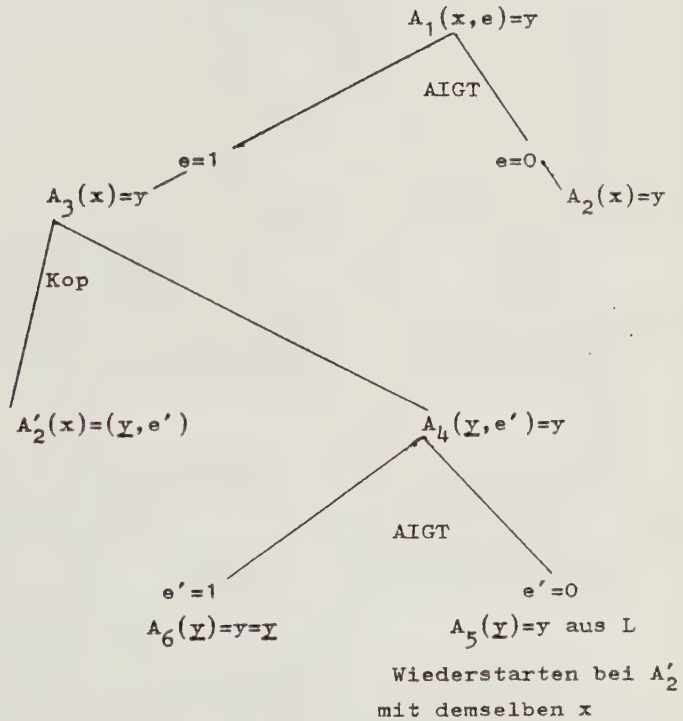
A) Ereignisbedingte Wiederholung

Eine Aufgabe A_1 wird dadurch erklärt, daß eine Vorschrift, die einer zweiten Aufgabe A_2 entspricht, beim Beobachten(bzw. Abtasten) eines Ereignisses e solange wiederholt(befolgt) wird, bis ein weiteres Ereignis e' auftritt:

-
- 1) Die Ereignisidee stammt aus Hamilton, M.u.a.: Higher Order..., a.a.O., S.16.
 - 2) Die folgenden Gebilde sind in Anlehnung an Arbeiten der eben zitierten Autoren entwickelt worden. Dabei ist zu bemerken, daß die Wiederstartgebilde von uns entwickelt worden sind und dort nicht gebraucht werden. Vgl. ebenda S.15 f sowie dieselben: Verification of an axiomatic Requirements Specification, Technical Report No. 10, Higher Order Software, Inc., Cambridge, MA, Oktober 1977, S.16.



a) zusammengefaßtes Gebilde



b) Vollständiges Gebilde

Abb.21. Ereignisbedingte Wiederholung

Das Auftreten bzw. Nichtauftreten der Ereignisse wird durch ' $=1$ ' bzw. ' $=0$ ' symbolisiert. A'_2 entspricht A_2 und einer Duplizierung. e wird als Teilinput von A_1 aufgefaßt. e' ist dagegen Teiloutput von A'_2 . In beiden Fällen hat das jeweilige Ereignis mit A_2 zu tun und kann vor, während und nach der Befolgung der Vorschrift, die A_2 entspricht, auftreten.¹⁾ e' muß so aufgefaßt werden, daß das Wiederstarten (hier reines Wiederholen mit demselben x) mit Sicherheit irgendwann nicht mehr stattfinden kann.

Beispiele für e und e' sind:

- a) e ist ein Fehler (d.h. etwa ein unzulässiger Input bzw. Output) und $e'=1$ bedeutet, daß A'_2 bereits dreimal wiederholt worden ist (y bleibt aus L , also unbestimmt).
- b) e und e' sind Merkmale von x und oder y (keine Fehler!).
- c) e und e' stellen Geschehnisse in der Systemumgebung dar.
- d) e und e' stellen Überlegungen von Menschen dar, die die Vorschriftenbefolgung beobachten.

1) Man kann also sowohl ein Datum als Ereignis als auch ein Ereignis als Datum auffassen.

B) Ereignisbedingte Vorschriftenauswahl:

Die Aufgabe A_1 bedeutet die Auswahl einer der ihr untergeordneten Aufgaben aufgrund des Auftretens von Ereignissen:

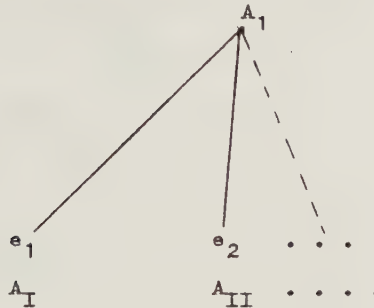


Abb. 22. Ereignisbedingte Vorschriftenauswahl

Es handelt sich hier um eine allgemeine Inputgruppenteilung mit mehr als zwei "Ästen" ¹⁾, wobei die Gruppenteilung aufgrund von Ereignissen erfolgt (die sich natürlich nicht überlappen).

1) Die Erweiterung von Gebilden um "Äste" kann am folgenden Beispiel erklärt werden:
 A_1 besteht aus A_2 und A_3 ; A_2 besteht wiederum aus A_I und A_{II} ; A_3 besteht aus A_{III} und A_{IV} so daß A_1 aus A_I bis A_{IV} besteht (dabei erfolgt die Aufgliederung nach einem einzigen Gebilde, z.B. AIGT).

C) Konstanteneinschub:

Eine Konstante wird aufgrund von menschlichen Überlegungen (Ereignis) in die Befolgung einer Vorschrift eingeschoben :

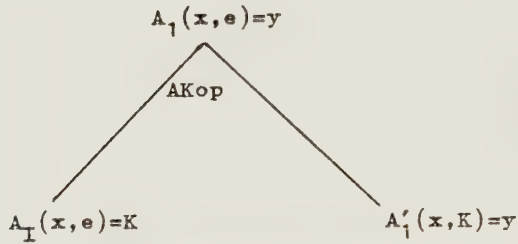
a) Vollständiges Gebildeb) Zusammengefaßtes Gebilde

Abb. 23. Das abgeleitete Gebilde 'Konstanten-
einschub'

4.6. Routinen

Die folgenden häufig verwendbaren Vorschriften können aus Elementarfunktionen unter Anwendung der Kopplung , Verkettung und Minimalisierung konstruiert werden. ¹⁾ Sie werden jeweils durch R symbolisiert und können als Vorschriften benutzt werden, ohne daß man ihre Ableitung aus Elementarfunktionen angibt. Manche Datenmaschinen enthalten Einrichtungen zur Befolgung solcher Vorschriften, die feste bzw. leicht einbaubare Bestandteile jener Maschinen darstellen:

$R(x,y) = x+y$ (Summe zweier Zahlen)

bzw. $= x-y$ (Unterschied zwischen Zahlen)

bzw. $= x.y$ (Produkt)

bzw. $= \frac{x}{y} + r$ (Division mit Rest r, $y \neq 0$)

$R(x) = \sqrt{x}$ (Quadratwurzel)

bzw. $= x^2$ (Quadrat)

$R(x,y) = \sum x.y$ (Summe der Produkte)

bzw. $= \prod (x+y)$ (Produkt der Summen)

bzw. $= \sum \sqrt{x^2 + y^2}$ (Summe der Quadratwurzeln aus
etc. der Quadrate-Summe)

1) Vgl. Heidler, K.u.a.: Rekursive Funktionen, a.a.O., S.50-92 und S.122 ff.

$$\begin{aligned}
 R(x_k) &= x_{k+m} && (m\text{-ter Nachfolger}) \\
 &\text{bzw.} = x_{k-m} && (m\text{-ter Vorgänger}) \\
 R(X) &= x_{\min} && (\text{Input mit der kleinsten Nummer} \\
 &&& \text{bzw. die kleinste Inputzahl}) \\
 R(X) &= x_{\max} && (\text{Input mit der größten Nummer} \\
 &&& \text{bzw. die größte Inputzahl})
 \end{aligned}$$

etc.

Unter Anwendung komplexerer Gebilde (z.B. abgeleiteter Gebilde) kann man für bestimmte Aufgabenbereiche jeweils die häufig zu verwendenden Vorschriften als Routinen auffassen und ohne Angabe ihrer Ableitung aus Elementarfunktionen und/oder weiteren Routinen einsetzen .

Die letzten vier Routinen(siehe oben) beschränkten sich auf jeweils einer Datengruppe X. Bisher haben wir wenig über den Aufbau von Datengruppen gesagt. Unter Berücksichtigung der vorhergehenden Bemerkung und der Erkenntnisse der Abschnitte 4.4 und 4.5 wollen wir uns im folgenden dem Aufbau von Datengruppen aus der Sicht eines Vorschriftengestalters widmen.

4.7. Besondere Vorschriften zum Aufbau von Datengruppen

Daten sollten im Hinblick auf die Objekte bzw. Objekte-Kombinationen, die sie symbolisieren, plausibel erscheinen. Dazu kann man in der Regel Plausibilitätsbedingungen als Vorschriften formulieren, deren Inputs die zu überprüfenden Daten und deren Output entweder "Bedingung erfüllt" oder "Bedingung nicht erfüllt" sind. Symbolisiert x z.B. eine Autonomnummer, so sollte es mit einem Buchstaben beginnen. Vorschriften, die der Überprüfung solcher Plausibilitätsbedingungen dienen, nennen wir Objektvorschriften.

Daten sollten auch im Hinblick auf die Vorschriften, in denen sie als Inputs bzw. Outputs vorkommen, plausibel erscheinen. Dazu formuliert man analog der obigen Überlegungen Datenvorschriften, die der Überprüfung der Datenplausibilität im Zusammenhang mit den zugehörigen Vorschriften dienen. Für $R(x)=x^2$ sollte man z.B. nachprüfen, ob x^2 eine nicht-negative Zahl ist.

Sowohl Objekt- als auch Datenvorschriften sind anhand der in diesem Kapitel beschriebenen hierarchischen Aufgliederung zu konstruieren, wenn sie von Datenmaschinen befolgt werden sollten.

Objekt- und Datenvorschriften sind nicht nur Plausibilitätsbedingungen, sondern auch ein Mittel zur Definition bzw. zum Aufbau von Datengruppen. Beispielsweise kann der Aufbau einer Datengruppe, deren Elemente geordnet sind (d.h. jedes Element hat einen Nachfolger und einen Vorgänger, mit Ausnahme des ersten und des letzten Elements) durch die Vorschrift $V_3(x_k) = x_{k+1}$ definiert werden. Im allgemeinen dient jede Vorschrift der Charakterisierung ihrer Inputs und Outputs.¹⁾

Andererseits ist jeder Umgang mit Daten als I-O-Umwandlung anzusehen. Die Manipulation von Datengruppen entspricht immer der Befolgung bestimmter Vorschriften. Wir definieren deshalb ein betriebliches Datenverwaltungssystem einfach als ein Mensch-Computer-System. Die zweckmäßige Aufgabenkonstruktion für ein solches System ist also nach der hier beschriebenen hierarchischen Aufgliederung bzw. nach einer äquivalenten Methode vorzunehmen.

1) Die Vollständigkeit dieser Charakterisierung soll hier nicht erörtert werden.

5. Vergleichende Wertung der ermittelten Aufgabenkonstruktionsgebilde

In Teil 2 haben wir festgestellt, daß die zweckmäßige Aufgabenkonstruktion für betriebliche Mensch-Computer-Systeme in der Erstellung von durch Datenmaschinen befolgbare Vorschriften besteht, wobei diese Befolgbarkeit in jedem Konstruktionsschritt zu gewährleisten ist. In Teil 3 haben wir die Elementarvorschriften angegeben, aus denen jede Vorschrift für eine Datenmaschine bestehen muß. Die Gebilde, mit deren Hilfe eine zweckmäßige Aufgabenkonstruktion, d.h. Zusammenfügung von Elementarvorschriften zu durch Datenmaschinen befolgbaren Vorschriften, erfolgen kann, sind in Teil 4 aus den Grundregeln der Konstruktion von rekursiven Funktionen abgeleitet worden. Neben diesen Regeln existieren nur normale Markov-Algorithmen und Turing-berechenbare Algorithmen als äquivalente Vorschriftsbegriffe für Datenmaschinen; letztere haben jedoch kein Konstruktionskonzept, so daß aus ihnen keine Aufgabenkonstruktionsgebilde abgeleitet werden können.¹⁾ Gäbe es ein viertes Vorschriftskonzept für Datenmaschinen, aus dem Konstruktionsgebilde abge-

1) Vgl. Adi, T.: Constructing Executable Functions for Men and Machines, Vortrag bei Higher Order Software, Inc., Cambridge, MA, November 1978, S.1 ff.

leitet werden könnten, so müßte seine Äquivalenz mit einem der genannten Vorschriftsbegriffe bewiesen werden, um die Konstruktionszweckmäßigkeit, d.h. insbesondere die Befolgbarkeit der Vorschriften durch Datenmaschinen(Turingmaschinen) zu gewährleisten. Dabei ist zu beachten, daß jene Gebilde jeden Aufgabekonstruktionsschritt bestimmen müssen. Bleiben einige Konstruktionsschritte ungeregelt (sei das auch nur vorläufig bzw. in einer Phase), so ist die Zweckmäßigkeit der Aufgabenkonstruktion nicht mehr gegeben. Werden etwa in einem Konstruktionsschritt nur die Datengruppen (Datenkonstruktion) oder nur die Umwandlungen(Funktionalonstruktion) betrachtet, so stellt das eine unzweckmäßige Aufgabenkonstruktion dar. Ein noch gröberer Verstoß gegen das Zweckmäßigkeitsprinzip ist die Aufbau-dann-Ablauf-Konstruktionsweise: in der Aufbauphase denkt man nicht an die Befolgbarkeit (d.h. den Ablauf). Die in Teil 4 ermittelten Aufgabenkonstruktionsgebilde bestimmen jeden Konstruktionsschritt und gewährleisten die Befolgbarkeit durch Datenmaschinen. In jedem Gebilde werden sowohl die Datengruppen als auch die Umwandlungen berücksichtigt.¹⁾

1) Das Problem der vollständigen Charakterisierung der Daten (vollständige Datengruppen- bzw. Datentypendefinition), das in Teil 4.7. angedeutet wurde, bleibt als erkenntnistheoretisches Anliegen außer Betracht, wenn hier die Vollständigkeit eines Aufgabenkonstruktionsschrittes erörtert wird.

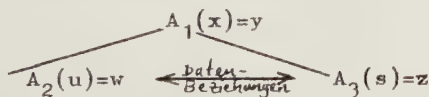
Die hierarchische Aufgliederung der Aufgaben in Vorschriften umschließt:

- a) die Herstellung der in Teil 2 definierten Symbol-Objekt-Verbindungen;
- b) die eigentliche Vorschriftenkonstruktion.

Die obigen Feststellungen führen zur Tatsache, daß die in Teil 4 ermittelten Aufgabenkonstruktionsgebilde eine zweckmäßige Aufgabenkonstruktion gewährleisten.

In Theorie und Praxis der Aufgabenkonstruktion für Mensch-Computer-Systeme geht man in der Regel von subjektiven Erfahrungswerten oder von empirischen Untersuchungen aus, wenn man Aufgabenkonstruktionsgebilde sucht. Im folgenden werden wir das beispielhaft belegen, wobei Verstöße gegen das Zweckmäßigkeitsprinzip und Kontraste zu den in Teil 4 ermittelten Gebilden angezeigt werden. Vorschriftsbegriffe, die eine Befolgbarkeit durch Datenmaschinen (Turingmaschinen) gewährleisten, sind nirgendwo in diesen Beispielen zu treffen.

Betrachten wir zunächst das HIPO-Gebilde:¹⁾



1) Vgl. Stay, J.F.: HIPO and Integrated Program Design, in: Freeman, P. und Wasserman, A.I. (Hrsg.): Software Design Techniques, a.a.O., S. 174 ff.

A_1 wird in A_2 und A_3 aufgegliedert, wobei die Daten-
gruppen X, Y, U, W, S und Z irgendwie zusammenhängen.
Das Gebilde stellt ein dem Ermessen überlassenes
Aufgliederungskonzept dar und läßt deshalb viele
Arten von Verstößen gegen das Zweckmäßigkeitsprin-
zip inform von durch Datenmaschinen unbefolgbar
Aufgabenhierarchien zu. Diese Behauptung läßt sich
anhand der Gebilde der strukturierten Programmie-
rung, die das HIPO-Gebilde zuläßt, anschaulich be-
legen: ¹⁾

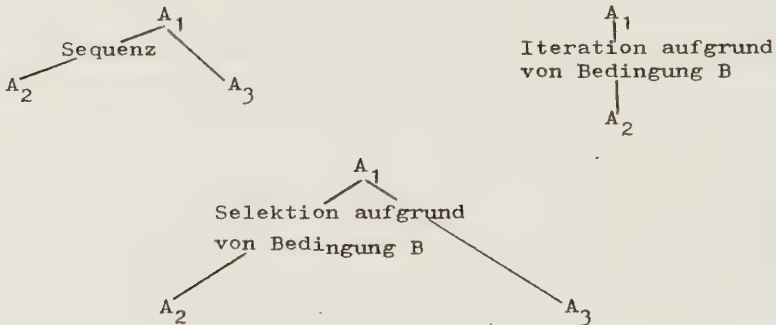


Abb. 24. Gebilde der strukturierten Programmierung

Bei der Sequenz wird die Reihenfolge der Befolgung
(A_2 dann A_3) ohne Berücksichtigung der Datengruppen-
struktur bestimmt. Dasselbe gilt für die Iteration
und die Selektion, so daß man hier von einer unzweck-
mäßigen Funktionalonstruktion sprechen kann.

1) Vgl. Jackson, M.: The Jackson Design Methodology,
in: Freeman, P und Wasserman, A.I. (Hrsg.): Soft-
ware Design Techniques, a.a.O., S. 219 f.

Ein Blick auf die Abbildungen 5, 13 und 14 läßt uns dagegen eine datengruppenstruktur-abhängige "Sequenz" erkennen. Eine datengruppenstruktur-abhängige "Selektion" ist die in Abb. 7 dargelegte Inputgruppen-
teilung bzw. die in den Abbildungen 15 und 16 veranschaulichte AIGT. Das Wiederstartgebilde in Abb. 7 stellt das Konzept einer bedingten "Iteration"^{dar}, wobei die Bedingung (das IGT-Gebilde) und die Iteration von der Datengruppenstruktur stark abhängen.

Die Structured-Design-Gebilde (SD-Gebilde) veranschaulichen wir wie folgt: ¹⁾

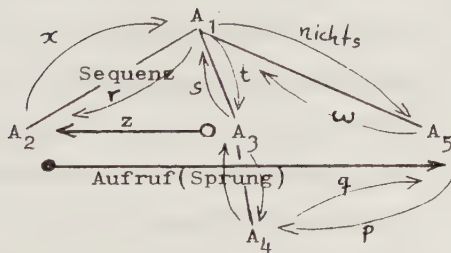


Abb. 25. Veranschaulichung der Structured-Design-Gebilde

1) Vgl. Stevens, W.P.; Myers, G.J. und Constantine, L.L.: Structured Design, in: Freeman, P und Waserman, A.I. (Hrsg.): Software Design Techniques, a.a.O., S. 247 ff.

In der obigen Abbildung sieht man die in den Originalgebilden durch Tabellen wiedergegebenen Datenbeziehungen durch Pfeile veranschaulicht. A_1 wird durch die Sequenz A_2 , A_3 und A_5 erklärt. Diese bekommen Inputs von A_1 und geben Outputs an A_1 . Außerdem bekommt A_2 einen Input z von ihrem Nachfolger in der Sequenz A_3 . A_4 (die Erklärung für A_3) tauscht Daten mit einer "Tante" in der Hierarchie aus (mit A_5). Die Datengruppenstruktur entspricht also nur teilweise der Aufgabenhierarchie. Mit anderen Worten: Eine teilweise Datenkonstruktion ist erlaubt.

Andererseits sind Sprünge (Aufrufe aus dem Inneren einer Aufgabe ins Innere einer anderen) wie hier von A_2 nach A_5 erlaubt, die eine teilweise (datengruppenstruktur-unabhängige) Funktionalonstruktion ermöglichen. Beides ist, wie oben erläutert, unzweckmäßige Aufgabenkonstruktion.

Die Jackson-Methodik ¹⁾ stellt einen primitiven Versuch dar, Funktional- und Datenkonstruktion zu einer zweckmäßigen Aufgabenkonstruktion zu vereinen. Dabei geht man jedoch phasenmäßig vor, so daß der Versuch von vorne herein zum Scheitern verurteilt wird. Jackson schlägt eine Funktionalonstruktion vor, die auf einer geteilten Datenkonstruktion aufgebaut wird. Die Teilung der Datenkonstruktion in Input- und Outputkonstruktion verschlechtert die Qualität (reduziert die Zweckmäßigkeit) der Aufga-

1) Vgl. Jackson, M.: The Jackson..., a.a.O., S. 221 ff.

benkonstruktion noch weiter. Jackson benutzt die für eine Funktionalonstruktion gedachten Gebilde der strukturierten Programmierung bei der geteilten Datenkonstruktion. Beispielsweise würde er also nach vollzogener Input- und Outputstrukturierung eine Inputiteration mit einer entsprechenden Outputiteration zu einer Programmiteration verschmelzen. Den drei Tätigkeiten der Input-, Output- und der verschmelzenden Programmstrukturierung entspricht hier die Benutzung eines Ket- bzw. Wiederstartgebildes, wobei man bedenken muß, daß die Verschmelzung problematisch sein kann (etwa wegen sogenannter structure clashes). Solche Probleme folgen aus der Konstruktionssplitterung (Daten-/Funktionalstruktur, Datendann-Funktionalstrukturierung, Input- und Outputstrukturierung getrennt), an der die hier ermittelten Gebilde nicht leiden, wie wir bei der Kritik der strukturierten Programmierung gezeigt haben.¹⁾

Aus sechs Axiomen, die sie aus empirischen Untersuchungen gefolgert haben, haben Hamilton und Zeldin dieselben in Teil 4 abgeleiteten Gebilde hergeleitet.²⁾ Die genannten Axiome gewährleisten an sich

1) Vgl. S. 48 f.

2) Vgl. Hamilton, M. und Zeldin, S.: Higher Order Software..., a.a.O., S.12 ff., S.20 und S.25 ff.

nicht die Befolgbarkeit durch Datenmaschinen. In den Teilen 4.3 und 4.4 haben wir die vier Grundgebilde unabhängig von diesen Axiomen aus den Regeln der Befolgbarkeit (rekursiver Funktionen) abgeleitet. Dadurch wurde die Zweckmäßigkeit dieser Gebilde der sogenannten Higher-Order-Software-(HOS-)Methodik bewiesen. Das stellt eine interessante Übereinstimmung zwischen empirischen und theoretischen Ergebnissen dar.

Im folgenden wollen wir versuchen, die Herkunft der Erscheinungen 'Funktional-/Datenkonstruktion' und 'Phasenkonzept' zu ergründen, zumal sie in den oben kritisierten Methodiken eine wichtige Rolle spielen. Bei den eingangs besprochenen fünf Versionen (ein Phasenkonzept) nähert man sich den Datenmaschinen schrittweise, wobei man in Wirklichkeit "Maschinen" verschiedener Abstufung schafft, denen die Versionen zugeordnet werden.¹⁾ Die eigentlichen Datenmaschinen werden also über Umwege angesprochen. Das erinnert uns an das Prinzip der Maschinenunabhängigkeit, nach dem die Aufgabenkonstruktion nicht in Abhängigkeit bestimmter Datenmaschinen erfolgen darf. Das Zweckmäßigkeitsprinzip besagt dagegen, daß die Befolgbarkeit (und damit die Berücksichtigung der befolgenden

1) Vgl. Hamilton, M. und Zeldin, S.: The Manager...,
a.a.O., S. 4

Datenmaschinen) zu jedem Konstruktionszeitpunkt gewährleistet sein sollte. Daraus ergibt sich der Verdacht, daß die angedeutete Maschinenabstufung im Phasenkonzept sowohl das Prinzip der Maschinenunabhängigkeit(durch Schaffung und Benutzung bestimmter "Maschinen")verletzt als auch durch eine Fehlinterpretation dieses Prinzips(inform einer beabsichtigten, jedoch nicht gelungenen Maschinenvermeidung) motiviert ist. Das Prinzip der Maschinenunabhängigkeit dient unseres Erachtens nur dem Ziel, unabhängig von den Datenmaschinen einer bestimmten Handelsmarke zu konstruieren. Mit wachsender Gemeinsamkeit zwischen den Handelsmarken artet dieses Prinzip jedoch in der eben kritisierten pauschalen Maschinenvermeidung aus.

Das Prinzip der Datenunabhängigkeit besagt, daß die Daten von den sie verarbeitenden Programmen (Vorschriften, Aufgaben) isoliert werden sollten, um den Aufwand für Programmänderung wegen anfallender Datenänderung gering zu halten.¹⁾ Auch hier ist eine Fehlinterpretation im Sinne einer Aufspaltung der Aufgabenkonstruktion in Daten- und Funktionalonstruktion denkbar.

Bei der Funktional-(bzw.Daten-)konstruktion werden die Daten (bzw. Umwandlungen) indirekt(z.B.implizite Umwandlungs- in der Datenkonstruktion bei Jackson), mangelhaft (z.B. SD-Gebilde) oder (datenstrukturmäßig)

1) Vgl. Wedekind, H.: Datenbanksysteme I, Zürich 1974, S.9

willkürlich (z.B. strukturierte Programmierung) strukturiert. Die Konstruktionsspaltung erfolgt auf Kosten der Zweckmäßigkeit.

Beim Streben nach Datenunabhängigkeit hat man Datenmanipulationssprachen eingeführt, die im Endeffekt Gebilde zur Konstruktion eines Teilbereiches der Aufgaben vorschreiben, die man in einem betrieblichen Mensch-Computer-System findet.¹⁾ Das bedeutet:

- a) daß Datenverwaltungssysteme eben betriebliche Mensch-Computer-Systeme sind, wie wir in Teil 4.7 bereits festgestellt haben;
- b) daß ein in die Datenkonstruktion abgeirrter Aufgabenkonstrukteur im Endeffekt zur simultanen Daten- und Funktionalonstruktion zurückkehren muß, wie das die Zweckmäßigkeit der Aufgabenkonstruktion erfordert.

Die Zweckmäßigkeit und die in Teil 2 festgestellte vorschriftengestalterische Eigenschaft des Anwenders "Mensch" in einem betrieblichen Mensch-Computer-System erfordern beharrlich eine transparente Computertechnologie (Hard- und Software).²⁾

1) Vgl. Wedekind, H.: Datenbanksysteme I, a.a.O., S. 108 ff.

2) Vgl. Hamilton, M. und Zeldin, S.: Higher Order..., a.a.O., S. 10 f.

6. Die Umwandlung von Konstruktionsgebilden in Befolungsgebilde

Betrachten wir noch einmal Abb.5 (S.19). Sie sagt uns im Grunde folgendes:

A_1 ist eine Kopplung von A_2 mit A_3 .

Will man jedoch A_1 als Vorschrift befolgen, so befolgt man nur A_2 und dann A_3 (als Vorschriften), ohne dabei an A_1 zu denken, es sei denn man möchte die Aufgabenstruktur 'im Auge behalten'.

Wir ordnen der Kopplung deshalb das folgende Befolungsgebilde zu:

$$A_2(x) = z \xrightarrow{\text{Kop}} A_3(z) = y \quad (\text{Kopplung})$$

Das Kop-Gebilde betrachten wir dagegen als Konstruktionsgebilde. Im folgenden werden in diesem Sinne alle bisher behandelten Gebilde als Konstruktionsgebilde betrachtet und in Befolungsgebilde umgewandelt.¹⁾

Der einfachen Spaltung ordnen wir das folgende Befolungsgebilde zu:

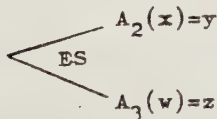
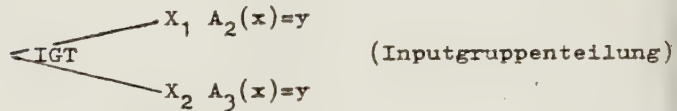


Abb. 26
(einfache
Spaltung)

1) Der Übergang von Konstruktions- zu Befolungsstrukturen entspricht in etwa dem von Programm-bäumen zu Programmablaufplänen (Flowcharts). Bislang war er nicht mechanisierbar. Hier wird ein solcher mechanischer Übergang dargelegt. Jackson geht manuell von Daten- zu Programmbäumen über.

Das Befolungsgebilde der Inputgruppenteilung ist das Folgende:



Dabei gilt: $X=(X_1, X_2)$.

Das Wiederstarten als Befolungsgebilde symbolisieren wir wie folgt (beispielhaft):

$A_1 \overset{\text{Kop}}{\dots} A_2 \overset{\text{IGT}}{\dots} A_3$ (ab A_1 mit x_{neu}) (Wiederstarten)

Die weiteren Gebilde sind alle abgeleitet und werden deshalb unter Verwendung der obigen vier Befolungsgebilde in ihre Befolungsformen umgewandelt. Das erfolgt jeweils durch Umwandlung der Grundgebilde, aus denen sie bestehen.

Beginnen wir mit der allgemeinen Kopplung (S.29). Wie bei der Konstruktion dieser Hierarchie sollten wir hier von oben nach unten (top-down) vorgehen.¹⁾

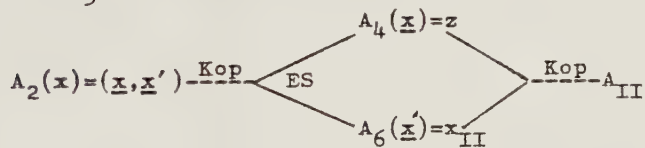
¹⁾ Diese Feststellung ist aufgrund des von uns benutzten Begriffes der hierarchischen Aufgliederung leicht zu verstehen.

Wir wandeln also zunächst die obere Kopplung um :

$$A_2(\underline{x})=(\underline{x},\underline{x}')\overset{\text{Kop}}{\dashrightarrow}A_3(\underline{x},\underline{x}')=(z,x_{II})\overset{\text{Kop}}{\dashrightarrow}A_{II} \quad 1)$$

$$A_{II}(z,x_{II})=y$$

Als Nächstes wird das A_3 untergeordnete Gebilde (einfache Spaltung) umgewandelt und an die Stelle von A_3 in die Befolgungskette eingefügt :



$$A_{II}(z,x_{II})=y$$

Da A_3 mit A_{II} gekoppelt ist, so werden die ES-Zweige "am Ausgang" zusammengefügt und mit A_{II} gekoppelt. Die ES-Verzweigung bedeutet, daß A_4 und A_6 unabhängig voneinander (gleichzeitig oder hintereinander) befolgt werden können. Schließlich wandeln wir das A_4 untergeordnete Gebilde (Kopplung) um und fügen es ein. Das Ergebnis ist in Abb. 27. angegeben.

1) Beim Erreichen des Zeilenendes wird nur die Aufgabenbezeichnung (ohne Daten) angegeben. Die Erklärung erfolgt in der Fortsetzungszeile.

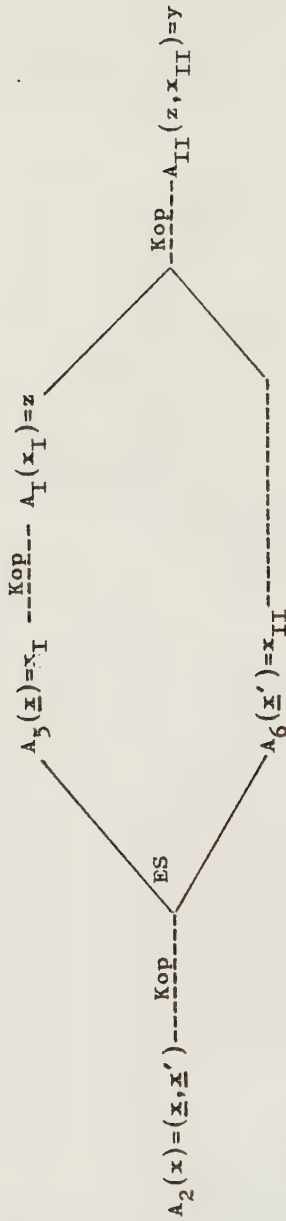


Abb. 27. Befolgungsgebilde zur allgemeinen Kopplung(vollständig)

x_I und x_{II} sind Teile von x , die sich überlappen können

Der allgemeinen Kopplung ordnen wir das folgende zusammengefaßte Befolungsgebilde zu:

$$\dots(x, \dots) \xrightarrow{\text{Kop}} A_I(x_I) = z \xrightarrow{\text{AKop}} A_{II}(z, x_{II}) = y$$

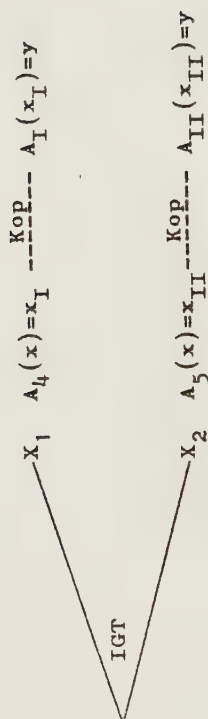
Abb.28. Befolungsgebilde zur allgemeinen Kopplung(zusammengefaßt)

Die Abbildung ist so zu interpretieren, daß eine Aufgabe mit dem Output x mit A_I gekoppelt ist. Die Punkte neben x und die Verzweigungspunkte deuten auf eine mögliche Verzweigung (ES bzw. IGT) hin.

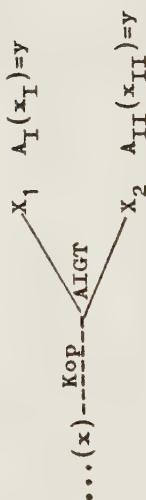
In den folgenden Abbildungen sind für die übrigen abgeleiteten Gebilde jeweils ein vollständiges und ein zusammengefaßtes Befolungsgebilde angegeben, die wir nach der obigen Prozedur abgeleitet haben.

Dabei erhält das allgemeine Wiederstartgebilde die folgende zusammengefaßte Befolungsform:

(a. ab A_2 mit x_{neu})

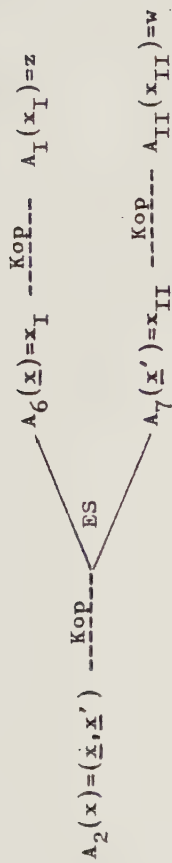


a) vollständig



b) zusammengefaßt

Abb. 29. Befolgungsgebilde zur allgemeinen Inputgruppenteilung
 $X=(X_1, X_2)$, x_I und x_{II} sind Teile von x , die sich überlappen können



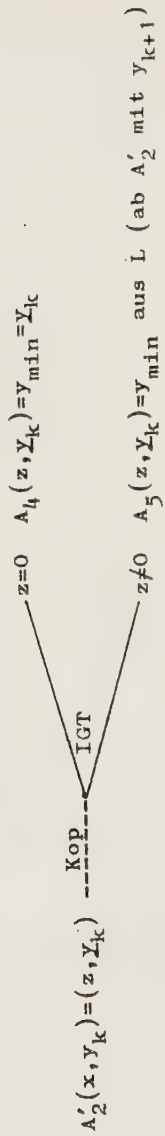
a) vollständig



b) zusammengefaßt

Abb. 30. Befolgungsgebilde zur allgemeinen einfachen Spaltung

x_I und x_{II} sind Teile von x , die sich überlappen können

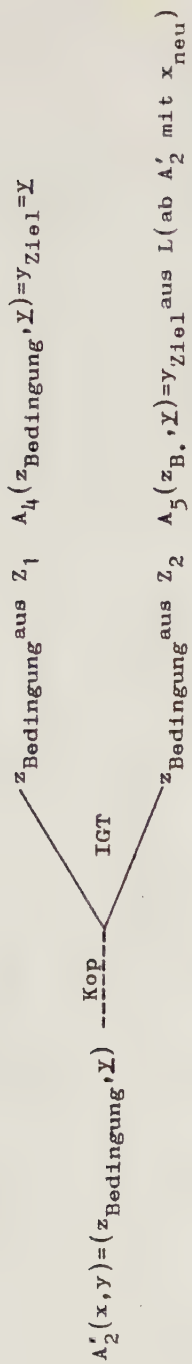


a) vollständig

$$A_2(x, y) = z \quad \text{Min} \quad A_1(x) = y$$

b) zusammengefaßt

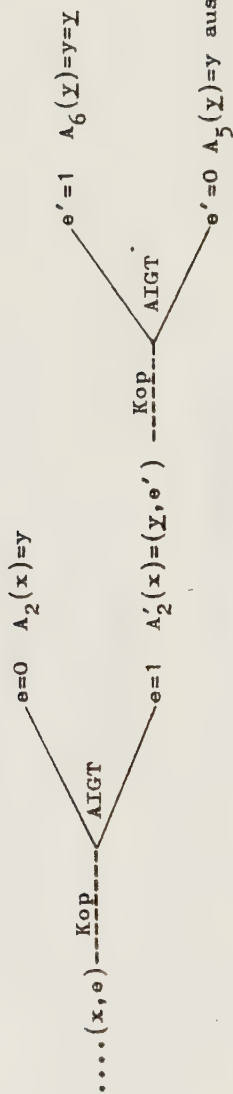
Abb. 31. Befolgungsgebilde zur Minimalisierung



a) vollständig

$$A_2(x,y)=z_{\text{Bedingung}} \quad \text{---Test---} \quad A_1(x)=y_{\text{Ziel}}$$

b) zusammengefaßt



a) vollständig

falls e wieder A_2
 $A_1(x, e) = y$ ----- $A_2(x) = y$
 bis e'

(64)

b) zusammengefaßt

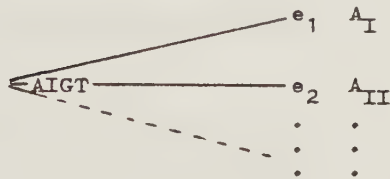
Abb. 33. Befolgungsgebilde zur ereignisbedingten Wiederholung

y ist eine vorläufige Kopie aus y

Die Verkettung (als Kopplungskette) erhält die folgende zusammengefaßte Befolungsform:

$$A_2(x_1)=y_1 \xrightarrow{\text{Ket}} A_3(x_k, x_{k-1}, y_{k-1})=y_k$$

Die ereignisbedingte Vorschriftenauswahl erhält als allgemeine Inputgruppenteilung die folgende zusammengefaßte Befolungsform:



Schließlich erhält der Konstanteneinschub die folgende Befolungsform, die einer allgemeinen Kopplung entspricht :

$$A_1(x)=y \xrightarrow{\text{Konstante}} A'_1(x,K)=y$$

7. Praxisnahe Beispiele

Wir betrachten im folgenden ein vereinfachtes Beispiel einer Auftragsabwicklungsaufgabe, die durch Menschen und/oder Datenmaschinen nach ihrer Umwandlung in eine Vorschrift befolgt werden soll. Zunächst betrachten wir die Aufgabenkonstruktion(hierarchische Aufgliederung in Vorschriften). Anschließend behandeln wir die Befolungsstruktur anhand von Befolungsgebilden.

7.1. Die Aufgabenkonstruktion

Gegeben seien Aufträge zur Lieferung und Fakturierung. Ein Auftrag bestehe aus einer Auftragsnummer(der die Nummer des Auftraggebers entnommen werden kann) und einer Mengen-Preise-Tabelle, etwa

Menge 1	Preis 1 (Stückpreis)
Menge 2	Preis 2
.	.
.	.

Die Fakturierung erfolge nach einer Durchsicht der Aufträge, bei der eine Lieferentscheidung (ja oder nein) getroffen wird.

Das Ergebnis der Fakturierung eines zu liefernden Auftrages bestehe in einer Rechnung und einem offenen Posten.

Die Lieferentscheidung werde aufgrund von einer menschlichen Überlegung getroffen, wobei die Auftragsdaten herangezogen werden.

Den obigen Überlegungen entspricht die folgende allgemeine Kopplung (siehe Abbildung 34) .

Betrachten wir nun die Aufgabe 'Durchsicht' näher. Die Entscheidung , ob geliefert werden sollte oder nicht, hänge von

a) dem Brutto-Rechnungsbetrag ($\sum_i \text{Menge}_i \cdot \text{Preis}_i$) und

b) der persönlich einzuschätzenden Zahlungsfähigkeit des Auftraggebers

ab, wobei b) nur für große Brutto-Rechnungsbeträge gilt. Für kleine Aufträge gelte der Grundsatz, daß sie in den "Zeitlücken" zwischen den großen geliefert werden.¹⁾ Die 'Durchsicht' erfolge täglich und die Entscheidung über kleine Aufträge bleibe solange bei 'nein', bis eine Zeitlücke erscheint.

Abb.35 veranschaulicht diese Überlegungen.

1) Die Auftragsgröße entspreche der Höhe des Brutto-Rechnungsbetrages. Die Grenze liege etwa bei 10000 (Geldeinheiten).

Auftragsabwicklung(Auftragsnummer,Mengen-Preise-Tabelle)=(Rechnung,offener Posten]



AKop

Durchsicht(Auftragsnummer,Mengen-Preise-Tabelle)=Lieferentscheidung

Fakturierung(Lieferentscheidung,Auftragsnummer,
Mengen-Preise-Tabelle)=(Rech.,o.Posten)

Abb. 24. Auftragsabwicklung als allgemeine Kopplung der Aufgabe 'Durchsicht' mit der Aufgabe 'Fakturierung'

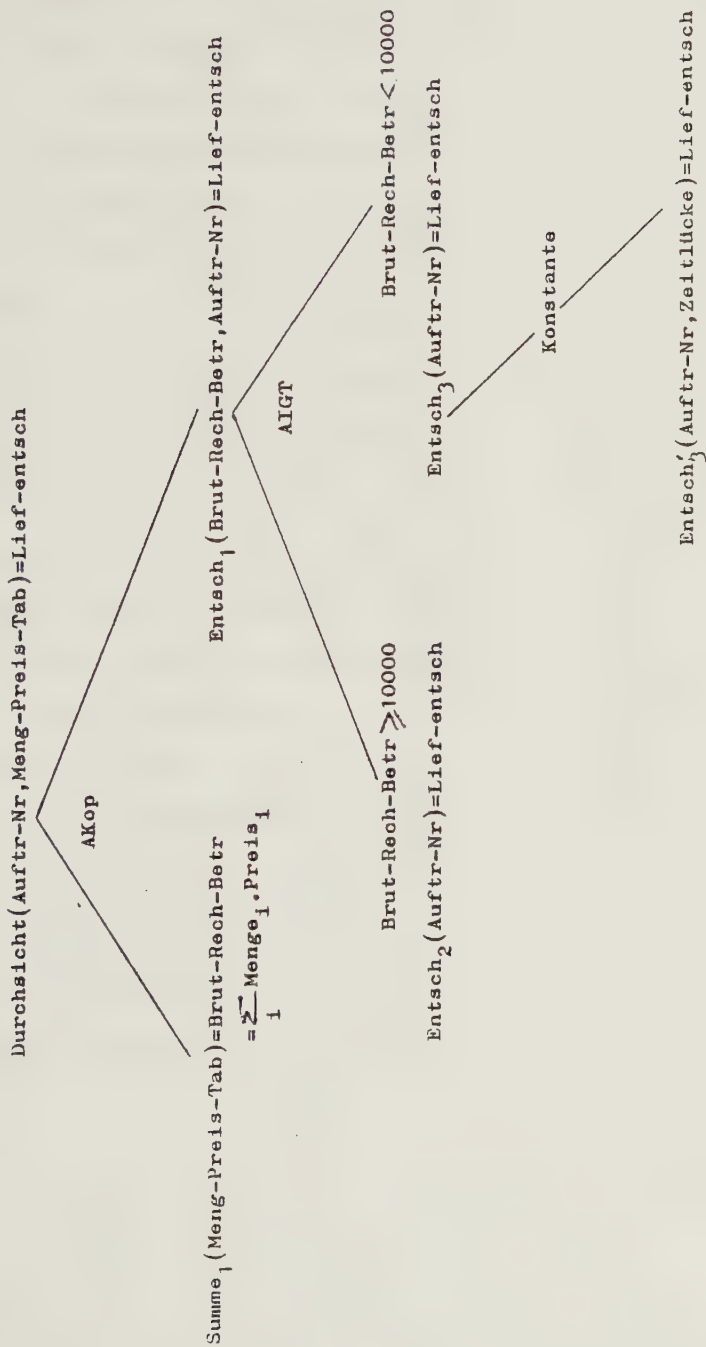


Abb. 35. Aufgliederung der Aufgabe 'Durchsicht'

In Abbildung 35 ist eine anschauliche Abkürzung der Begriffe vorgenommen worden. Summe₁ stellt eine Routine dar. Zeitlücke ist eine Konstante, die aufgrund eines betrieblichen Ereignisses bestimmt wird. Entscheidungsaufgabe 2 (Entsch₂) ist eine menschliche Ermessensentscheidung.

Betrachten wir nun die 'Zeitlücken' näher.

Abbildung 36 veranschaulicht die vollständige Form des Konstanteneinschubes. Ereignis 1, das in der Zusammenfassung nicht erscheint, soll wie folgt interpretiert werden:

'Zeitlücke' sei eine Entscheidung mit "ja" oder "nein" aufgrund der Durchsicht einer Tabelle, die aus Zeitintervallen besteht und mit dem Eintreffen von Ereignis 1 zur Verfügung gestellt wird. Ein Mensch schätzt die zur Lieferung des gegebenen Auftrages (laut Auftr-Nr) benötigte Zeitdauer (geschätzte Auftragszeit genannt) ab. Er sucht dann aufgrund dessen in der zur Verfügung gestellten Tabelle nach dem ersten passenden Zeitintervall. Diese Suche veranschaulichen wir durch die Testschleife in Abb. 37.

Entsch₃(Auftr-Nr, Ereignis 1)=Lief-entsch

AKop

Entsch₄(Auftr-Nr, Ereignis 1)=Zeittücke

Entsch'₃(Auftr-Nr, Zeittücke)=Lief-entsch

Abb. 36. Der vollständige Konstanteneinschub

Zeittücke= ja oder nein

Entsch₄(Auftr-Nr, Ereignis 1 d.h. Zeitintervalltabelle)=Zeitlücke

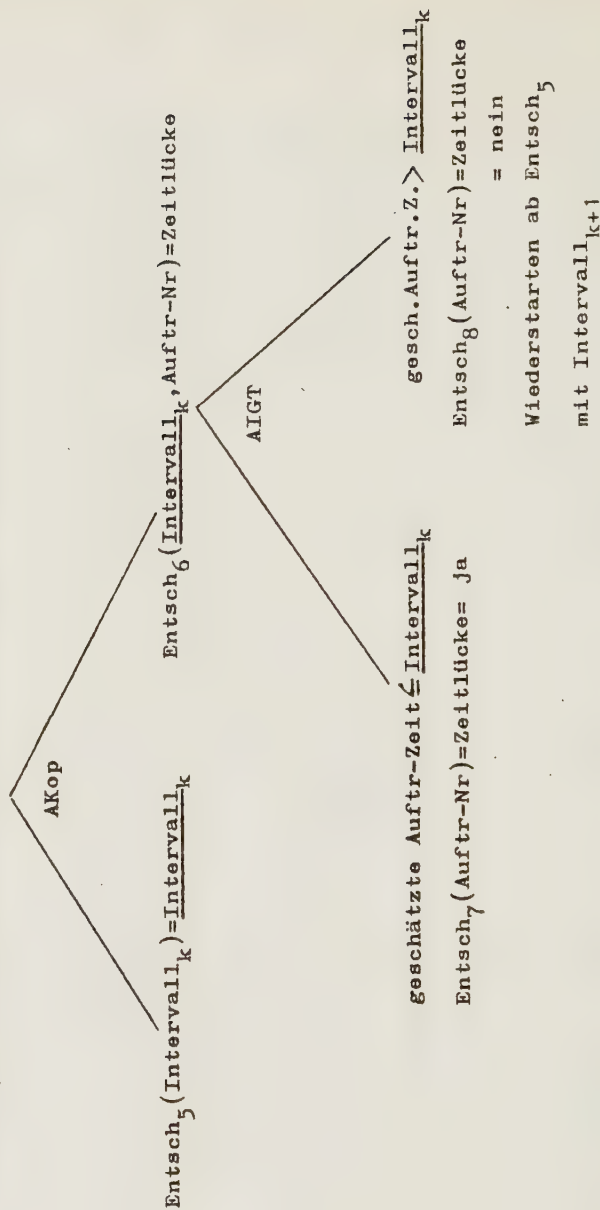


Abb. 37. Testschleife zur Zeitlückenermittlung

Die Aufgabe "Fakturierung" soll nun aufgegliedert werden.

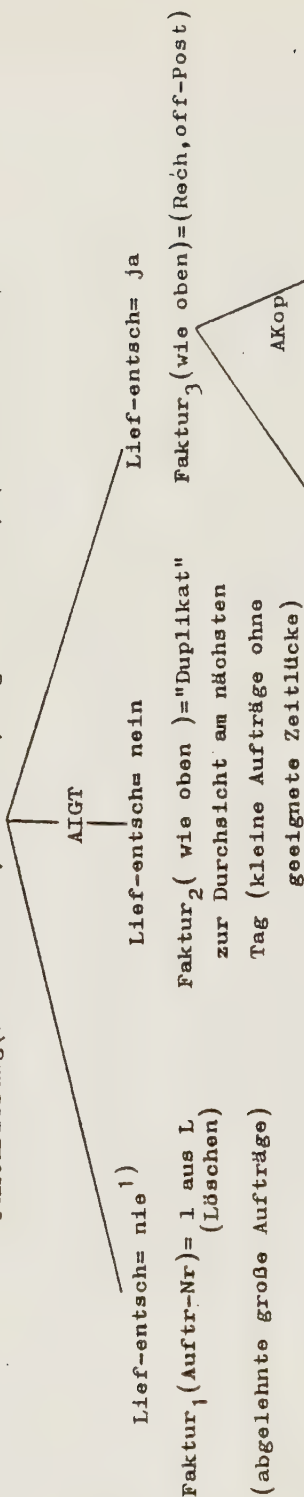
Erfolgt eine Lieferung (Lief-entsch= ja), so wird eine Netto-Betragsrechnung durchgeführt. Anschließend werden Rechnungen gedruckt und eine Liste der offenen Posten erstellt, wobei wir davon ausgehen, daß die Fakturierung erst nach Beendigung der "Durchsicht" (für alle Aufträge) erfolgen sollte. (Siehe Abb. 38)

Die Netto-Betragsrechnung erfolgt unter Berücksichtigung des Nachlasses für bestimmte Auftraggeber (die Inputgruppe der Auftr-Nr wird in zwei Teile aufgespaltet). Der Nachlaß beträgt 10 %. Abbildung 39 veranschaulicht die Aufgliederung der Netto-Betragsrechnung.

Betrachten wir schließlich je ein Beispiel für Objektvorschriften und Datenvorschriften für die Plausibilität der behandelten Aufträge. Abbildung 40 veranschaulicht eine Objektvorschrift (Mengen dürfen nicht ≤ 0 sein).

In Abb. 41 ist eine Datenvorschrift aufgegliedert (Aufträge, die für den nächsten Tag dupliziert worden sind, dürfen keine großen sein; das stellt eine Überprüfung des Outputs im Hinblick auf eine Vorschriftengruppe dar).

Fakturierung(Lief-entsch,Auftr-Nr,Meng-Preis-Tab)=(Rech,off-Post)



(74)

Net-Betr-Rech(Auftr-Nr,Meng-Preis-Tab)=Net-Betr

Faktur₄(Net-Betr,Auftr-Nr,Meng-Preis-Tab)=
(Rech,off-Pos)

AES

Rech-Druck(alles)=alles O-P-List(Net-Betr)=
=Rech Net-Betr=off-Post

Abb. 38. Fakturierung(ohne Aufgliederung der

Net-Betr-Rech)

$$\text{Net-Betr-Rech}(\text{Auftr-Nr}, \text{Meng-Preis-Tab}) = \text{Net-Betr}$$

AKop

$$\text{Nachl-Rech}(\text{Auftr-Nr}) = \text{Nachl-Faktor}$$

$$\begin{aligned} \text{Net-Betr-Rech}_1(\text{Meng-Preis-Tab}, \text{Nachl-Faktor}) &= \text{Net-Betr} \\ &= \text{Nachl-Faktor} \cdot \sum_i \text{Menge}_i \cdot \text{Preis}_i \end{aligned}$$

IGT

Gruppe 1

Gruppe 2

$$\text{Nachl}_1(\text{Auftr-Nr}) = 1$$

$$\text{Nachl}_2(\text{Auftr-Nr}) = 0,9$$

(75)

Abb. 39. Aufgliederung der Netto-Betragsrechnung

Objektvorschrift (Auftr-Nr, Meng-Preis-Tab) = o.k./nicht o.k.

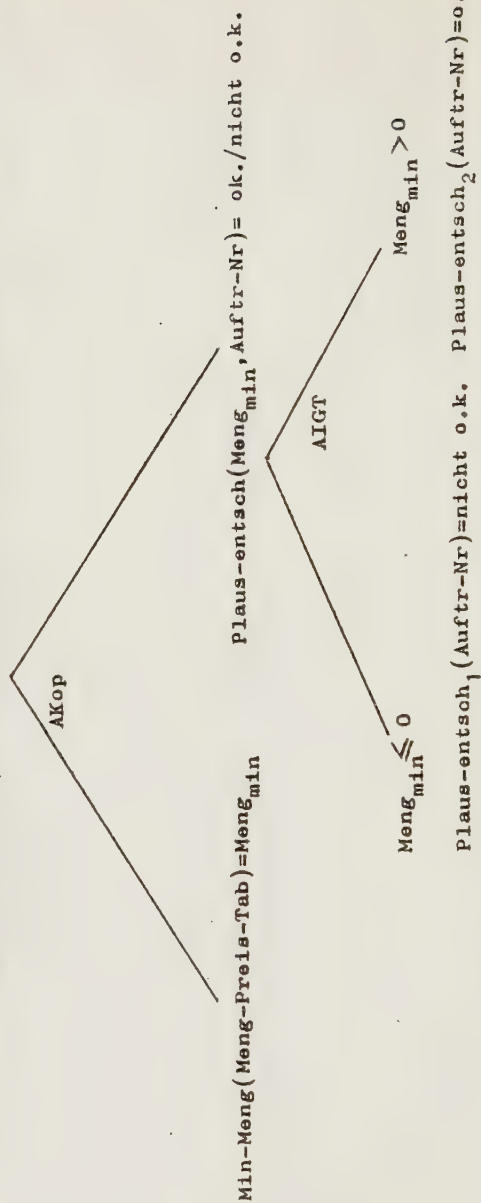


Abb. 40. Eine Objektvorschrift

Datenvorschrift (Duplikat, d.h. : $\text{Auftr-Nr}, \text{Meng-Preis-Tab}$) = o.k. / nicht o.k.

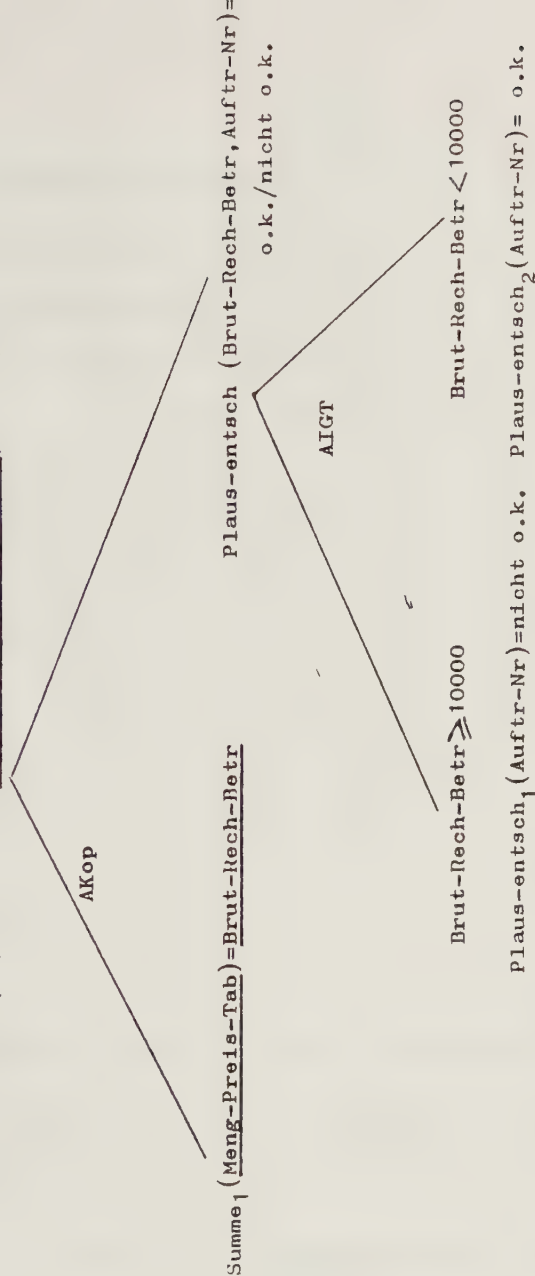


Abb. 41. Eine Datenvorschrift

7.2. Befolungsstrukturen

Aufgrund der Erkenntnisse von Kapitel 6 entwickeln wir im folgenden Befolungsstrukturen für alle in Abschnitt 7.1 besprochenen Aufgabenhierarchien.

Die folgenden Abbildungen entsprechen den Abbildungen 34 bis 41 in derselben Reihenfolge, wobei man berücksichtigen muß, daß die Auftragsabwicklung aus mehreren zusammenhängenden Aufgaben besteht, die in getrennten Abbildungen aufgliedert worden sind.

Durchsicht(Auftr-Nr,Meng-Preis-Tab)=Lief-entsch ^{AKop}-----Fakturierung
 Fakturierung(Lief-entsch,Auftr-Nr,Meng-Preis-Tab)=(Rech,off-Post)

Abb. 42. Befolungsstruktur zu Abb.34

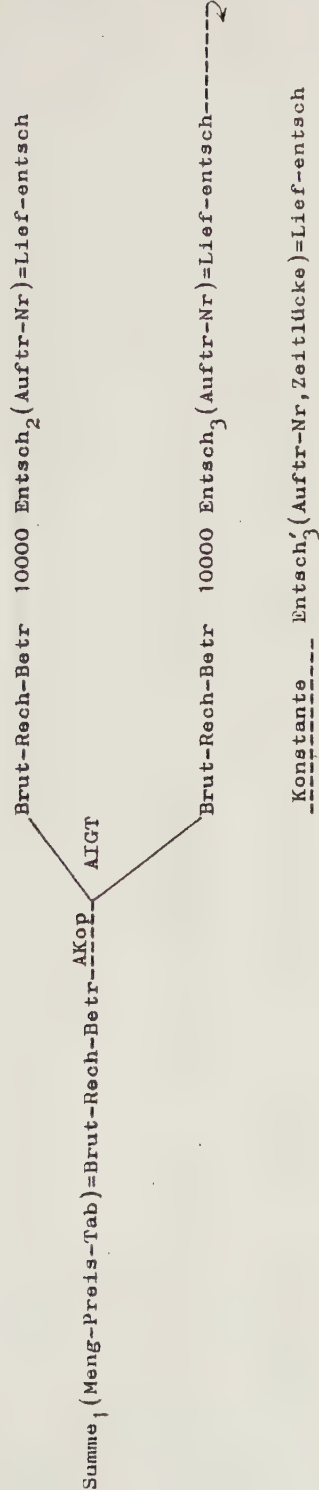


Abb. 43. Befolungsstruktur zu Abb. 35

Entsch₄(Auftr-Nr, Ereignis 1)=Zeitlücke-AKop----- Entsch'₃(Auftr-Nr, Zeitlücke)=Lief-entsch

a) zu Abb. 36

gesch. Auftr-Zeit Intervall_k
Entsch₇(Auftr-Nr)=Zeitlücke= ja

Entsch₅(Intervall_k)=Intervall_k AKop----- AIGT

gesch. Auftr-Zeit Intervall_k

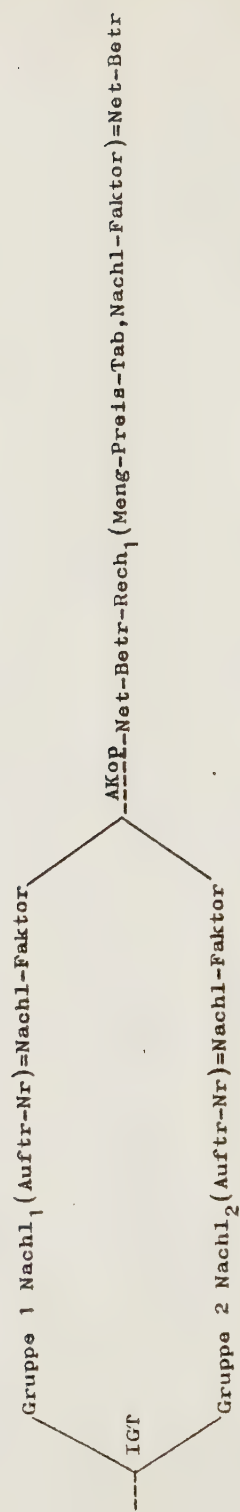
Entsch₈(Auftr-Nr)=Zeitlücke= nein(a.ab Entsch₅ mit Intervall_{k+1})

b) zu Abb. 37

Lief-entsch=nie Faktur₁(Auftr-Nr)= 1 aus L

-----ALGT---Lief-entsch= nein Faktur₂(Auftr-Nr, Meng-Preis-Tab)=(Auftr-Nr, Meng-Preis-Tab)

Lief-entsch= ja Net-Betr-Rech(Auftr-Nr, Meng-Preis-Tab)=Net-Betr AKop ----- AES
Rech-Druck(Net-Betr, Auftr-Nr, M.-P.-Tab)=Rech
O-P-List(Net-Betr)=off-Post



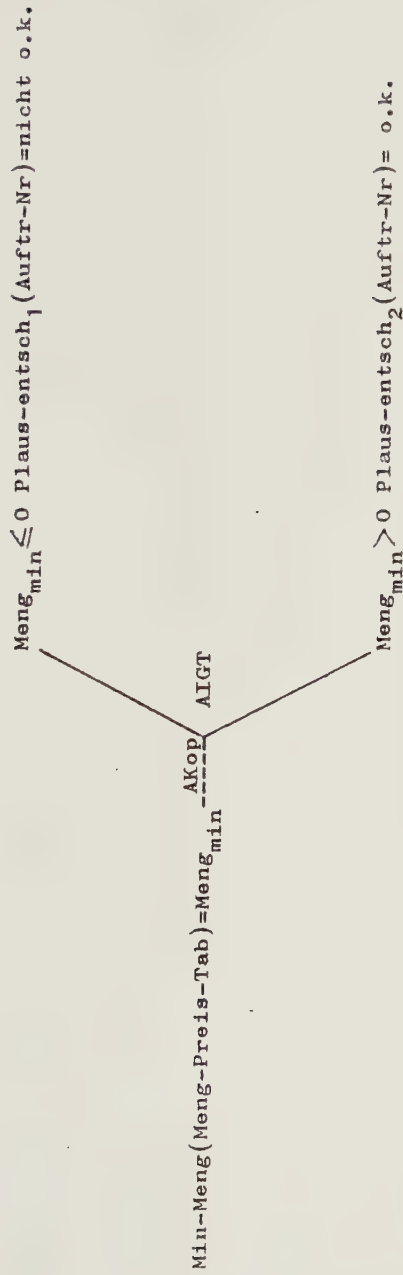


Abb. 47. Befolungsstruktur zu Abb. 40

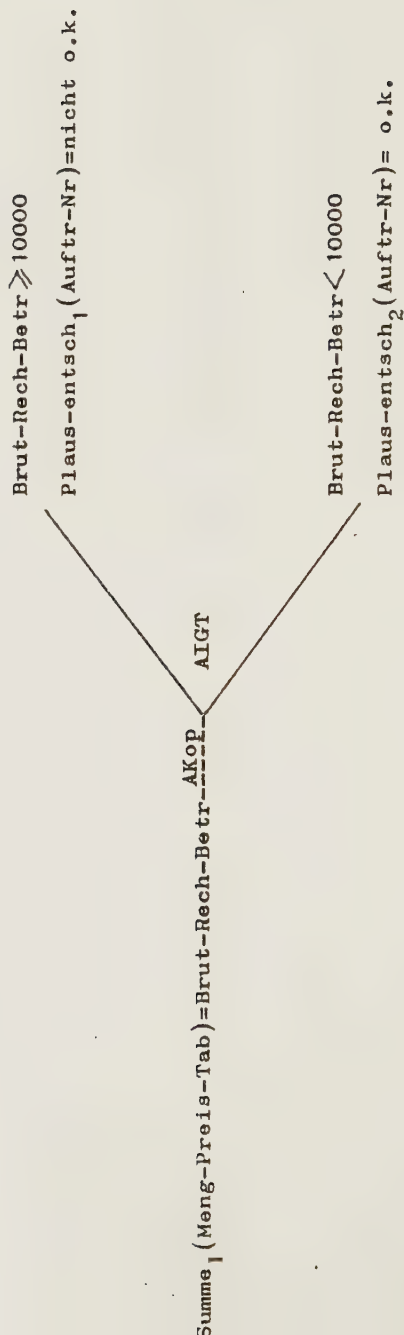


Abb. 48. Befolungsstruktur zu Abb. 41

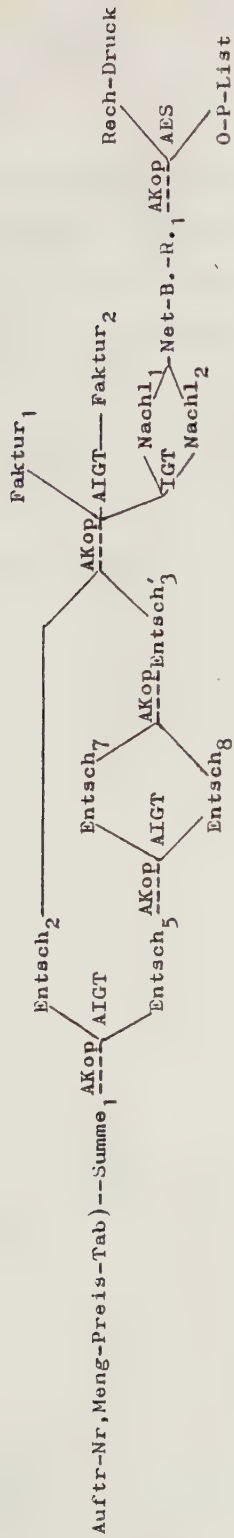


Abb. 49. Zusammengefaßte Befolungsstruktur der Auftragsabwicklung

In Abbildung 49 ist die Befolungsstruktur der Auftragsabwicklungsaufgabe zusammengefaßt dargestellt. Diese Zusammenfassung ist aus der Zusammenfassung der Aufgabenhierarchien der Abbildungen 34 bis 39 abgeleitet worden (siehe Abb. 50), wobei wir von oben nach unten vorgehend jedes Konstruktionsgebilde durch das jeweilige Befolungsgebilde ersetzt haben. Dabei nahmen wir die Abbildungen 42 bis 46 zu Hilfe.

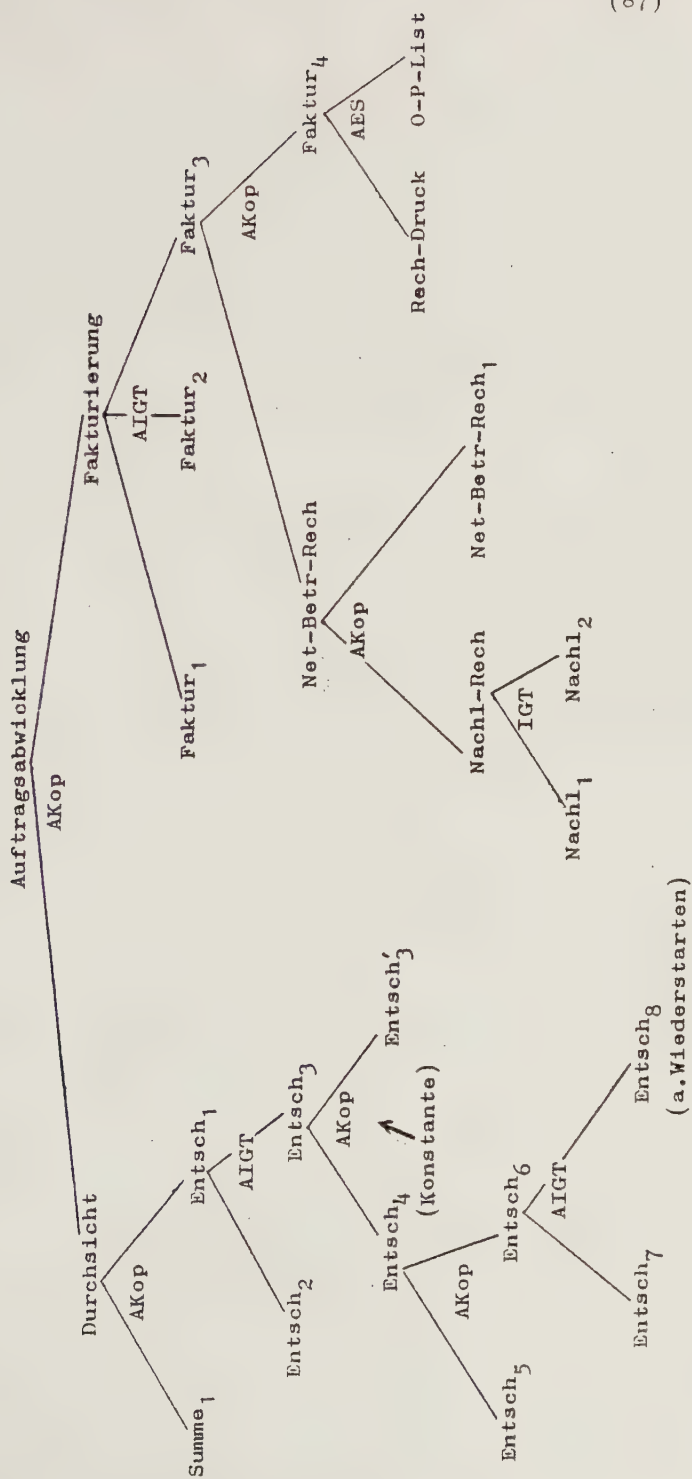


Abb. 50. Zusammengefaßte Aufgliederung der Auftragsabwicklung

8. Die Erkenntnisse der Arbeit

Das Ziel unserer Arbeit ist das Auffinden einer zweckmäßigen Aufgabenkonstruktionsweise für betriebliche Mensch-Computer-Systeme. Dabei bedeutet die Zweckmäßigkeit die Gewährleistung der Aufgabenerfüllung zu jedem Zeitpunkt der Konstruktion.

Wir haben in Kapitel 2 festgestellt, daß die eigentliche Aufgabenkonstruktionsarbeit des Menschen in der Vervollständigung von Aufgaben zu Vorschriften besteht. In der Suche nach Objekten für gegebene Symbole (Aufgaben) benutzt dabei der Mensch sein Erstbenennungsvermögen. Die Vorschriftenbefolgung als Aufgabenerfüllung haben wir den Datenmaschinen zugeordnet.

Aufgrund dieser Überlegungen haben wir Elementarteilchen für Datenmaschinenvorschriften (Elementarvorschriften) ermittelt, besteht doch jedes Objekt aus seinen Elementarteilchen und kann nur aus solchen konstruiert werden.¹⁾ Dieser Ermittlung haben mathematische Erkenntnisse über drei abstrakte Vorschriftsbegriffe, deren Äquivalenz bewiesen worden ist, zugrunde gelegen.

1) Vgl. Abschnitt 3.2

Der Zusammenhang der Elementarvorschriften in einer Vorschrift unterliegt ausschließlich drei Regeln der Kombination von mathematischen Funktionen, die wir als Vorschriften für Datengruppen ansehen.¹⁾ Wir haben festgestellt, daß die Anwendung dieser Regeln hierarchisch aufgebaut werden kann.²⁾ Eine tiefgehende Analyse der Regeln selbst hat ihren hierarchischen Aufbau gezeigt.³⁾ Wir haben aufgrund dieser Erkenntnisse Gebilde zur hierarchischen Aufgliederung von Aufgaben in Vorschriften entwickelt und (zum Teil anhand von Erkenntnissen der HOS-Methode) erweitert (durch abgeleitete Gebilde).⁴⁾ Anschließend haben wir Routinen als häufig verwendbare Vorschriften eingeführt.⁵⁾ Für die Überprüfung der Plausibilität von Datengruppen haben wir die Benutzung besonderer Vorschriften vorgeschlagen, die wie andere Vorschriften hierarchisch aufgebaut werden können.⁶⁾

1) Vgl. Abschnitte 4.1 und 4.2

2) Vgl. Abschnitt 4.3

3) Vgl. Abschnitt 4.3

4) Vgl. Abschnitte 4.3 , 4.4 und 4.5

5) Vgl. Abschnitt 4.6

6) Vgl. Abschnitt 4.7

Im Lichte der Erkenntnisse von Kapitel 4 haben wir in Kapitel 5 zu einschlägigen Aufgabenkonstruktionsansätzen und -prinzipien Stellung genommen. Dabei kamen wir zur Überzeugung, daß zwei bekannte Prinzipien (Maschinen- und Datenunabhängigkeit) die Entwicklung der Aufgabenkonstruktion für betriebliche Mensch-Computer-Systeme in die Unzweckmäßigkeit getrieben haben. Zum einen prägten sie das Phasenkonzept (Maschinenphasen bzw. Aufbau-dann-Ablauf). Zum anderen führten sie zur Aufspaltung der Aufgabenkonstruktion in Daten- und Funktionalonstruktion. Beides bedeutet eine Isolierung von zweckmäßigerweise nicht zu isolierenden Konstruktionsschritten. Aus dieser Sicht wurden die Gebilde der Ansätze HIPO, Jackson, strukturierte Programmierung und Structured Design analysiert und mit den in Teil 4 ermittelten Gebilden verglichen. Die Grundgebilde letzterer sind bereits aus empirisch begründeten Axiomen (der HOS-Methodik) abgeleitet. Erst durch ihre Ableitung aus den Regeln der Konstruktion rekursiver Funktionen haben wir ihre Zweckmäßigkeit (in Teil 4.3 und am Anfang von Teil 4.4.) bewiesen. Insbesondere haben wir den empirisch nicht zu liefernden Beweis der Befolgbarkeit durch Datenmaschinen erbracht.

Eine Vorschrift ist bei uns per Definition durch eine Datenmaschine (Turingmaschine) befolgbar.¹⁾ Befolgbare sind auch alle ermittelten Gebilde, da wir von der genannten Vorschriftseigenschaft ausgehen. Deshalb gelang es uns in Kapitel 6 ohne große Mühe, für jedes Konstruktionsgebilde ein Befolgungsgebilde zu finden. Der Übergang von Aufgabenstruktur zu Aufgabenerfüllungsstruktur ist somit eine "Routinearbeit" geworden, die man Datenmaschinen zuordnen kann.

Fazit : Man braucht nur eine Aufgabenversion, die hierarchisch und somit transparent ist und in allen Bestandteilen erfüllungs- bzw. befolgungsgewährleistend ist.

Unsere Arbeit hat somit ihr Ziel erreicht.

In Kapitel 7 haben wir die Erkenntnisse unserer Arbeit an praxisnahen Beispielen veranschaulicht. Dabei haben wir auch gezeigt, daß Vorschriften für Datenmaschinen unvollständige Vorschriften (Ermessensentscheidungen, Entscheidungen aufgrund von 'Ereignissen') enthalten können, die durch den menschlichen Eingriff zur Vollständig-

1) Vgl. Abschnitte 3.2 und 4.2 sowie Kapitel 2

keit und damit zur Befolgbarkeit durch Datenmaschinen gebracht werden können.¹⁾

Unsere Arbeit stellt die 5-Versionen-dann-Testen-Vorgehensweise in Frage und liefert einen Ersatz für sie, der einer zweckmäßigen Aufgabenkonstruktion im Sinne der betriebswirtschaftlichen Organisationstheorie entspricht.

Da die I-O-Umwandlung Gegenstand jeder Aufgabenkonstruktion (für jeden Betrieb) ist, kann die gefundene Vorgehensweise als Methode zur Aufgabenkonstruktion schlechthin betrachtet werden. Die "ewige" Konfrontation zwischen Aufbau- und Ablauforganisation wäre damit beseitigt;²⁾ nach unserem Ansatz 'laufen beide Hand in Hand'.

Unsere Erkenntnisse sind abstrakter Natur und von der vorhandenen Computer-Technologie unabhängig. Die hierarchische Aufbauweise, die Daten und Vorschriften umschließt, erlaubt eine transparente "Implementierung", wobei die Umgestaltung aufgrund der Baumartigkeit nur leicht übersehbare Folgen haben kann.

1) Wir haben dabei die Ergebnisse von Abschnitt 4.5.2 benutzt. Die Unvollständigkeit der Vorschriften ist hier "eingeplant".

2) Vgl. Witte, E.: Ablauforganisation, a.a.O., S.23

Verzeichnis der verwendeten Abkürzungen

AES	Allgemeine einfache Spaltung
AGIT	Allgemeine Inputgruppenteilung
AKop	Allgemeine Kopplung
ES	Einfache Spaltung
Hrsg.	Herausgeber
IEEE	Institute of Electrical and Electronics Engineers
IGT	Inputgruppenteilung
Ket	Verkettung
K	Konstante
Kop	Kopplung
L	leere Datengruppe
	"aus L" bedeutet "nicht vorhanden"
Min	Minimalisierung
Vol.	Volume
$y_{\min}$	das Datum mit der kleinsten Nummer
Y	die Datengruppe der y's
$Y_1, Y_2 \dots$	Teilgruppen aus Y
Σ	ein Duplikat von y
(z,w)	ein aus zwei Symbolen bestehendes Datum

Literaturverzeichnis

(Datum in Klammern bezieht sich auf die Erstveröffentlichung)

- Adi, T.: Constructing Executable Functions for Men and Machines,
Vortrag bei Higher Order Software, Inc., Cambridge, MA, November
1978
- Akademie für Organisation in Gießen (Hrsg.): Handlexikon Organi-
sation, Frankfurt am Main 1976
- Boehm, B.W.: Software Engineering, IEEE Transactions on Computers,
Vol. C-25, No.12, Dezember 1976, S. 1226-1241
- Brauer, W. und Indermark, K.: Algorithmen, rekursive Funktionen
und formale Sprachen, Mannheim 1968
- Freeman, P. und Wasserman, A.I.(Hrsg.): Software Design Techniques,
Long Beach, California 1976
- Grochla, E.(Hrsg.): Handwörterbuch der Organisation, Stuttgart 1969
derselbe (Hrsg.): Elemente der organisatorischen Gestaltung,
Hamburg 1978
- Hamilton, M. und Zeldin, S.: Higher Order Software- A Methodology
for Defining Software, IEEE Transactions on Software Engineering,
Vol. SE-2, No.1, März 1976, S.9-32
- dieselben: The Manager as an Abstract Systems Engineer, Technical
Report No.5, Higher Order Software, Inc., Cambridge, MA, Juni 1977
- dieselben: Verification of an Axiomatic Requirements Specification,
Technical Report No.10, Higher Order Software, Inc., Cambridge,
MA, Oktober 1977
- Heidler, K.; Hermes, H. und Mahn, F.K.: Rekursive Funktionen,
Zürich 1977
- Jackson, M.: The Jackson Design Methodology, in: Freeman, P. und
Wasserman, A.I.(Hrsg.): Software Design Techniques, Long Beach,
California 1976, S.219-234 (1975)
- Kosiol, E.: Ablauforganisation, in: Grochla, E.(Hrsg.): Handwörter-
buch der Organisation, Stuttgart 1969, S.172-191
- derselbe: Aufgabenanalyse und Aufgabensynthese, in: Grochla, E.
(Hrsg.): Elemente der organisatorischen Gestaltung, Hamburg
1978, S.66-84

- Meschkowski, H.: Mathematisches Begriffswörterbuch, Mannheim 1976
- Stay, J.F.: HIPO and Integrated Program Design, in: Freeman, P. und Wasserman, A.I.(Hrsg.): Software Design Techniques, Long Beach, California 1976, S.174-178 (1976)
- Stevens, W.P.; Myers, G.J. und Constantine, L.L.: Structured Design, in: Freeman, P. und Wasserman, A.I.(Hrsg.): Software Design Techniques, Long Beach, California 1976, S.247-255 (1974)
- Teichroew, D.: Improvements in the System Life Cycle, in: Freeman, P. und Wasserman, A.I.(Hrsg.): Software Design Techniques, Long Beach California 1976, S.37-43 (1974)
- Wedekind, H.: Datenbanksysteme I, Zürich 1974
- Witte, E.: Ablauforganisation, in: Grochla, E.(Hrsg.): Handwörterbuch der Organisation, Stuttgart 1969, S.20-30

LEBENS LAUF

Am 27. August 1950 wurde ich als Sohn des Angestellten Anoir Adi und seiner Ehefrau Nadima, geb. Tahhan in Damaskus geboren.

Nach dem Besuch verschiedener Schulen in Hama und Aleppo bestand ich im Sommer 1968 das syrische Abitur.

Anschließend studierte ich für zwei Jahre Elektrotechnik an der Universität Aleppo.

Mein Studium der Wirtschaftswissenschaften begann in Göttingen im Wintersemester 1971/72 und endete dort am 5. November 1975 mit der Diplomprüfung für Betriebswirte.

Seit dem Wintersemester 1976/77 bin ich an der Johannes Gutenberg-Universität immatrikuliert.

Am 7. Februar 1979 bestand ich die mündliche Doktorprüfung.

Mainz, den 13.2.1979

Tammam Aeli

